An Evaluation of

Software Process Modelling

in Practice

Keith Thomas Phalp BSc, PGCE, MSc

In part fulfilment of the degree of Doctor of Philosophy

Acknowledgements

Special thanks to Martin Shepperd for making, first software metrics, and then process modelling appear to be interesting and worthwhile areas of study, for believing in the worth of my ideas, for his understanding and sympathetic supervision, for the many words of wisdom, and mostly for being a good friend. Thanks to Darrel Ince for his enthusiasm for my work, his support and his useful comments.

Thanks to Shari Lawrence Pfleeger, for her many efforts to turn my thesis from a rambling piece of prose, into a piece of science.

Thanks to my colleagues at Bournemouth University, especially to Steve Webster for his inspiration and enthusiasm for understanding software engineering, and to Liguang Chen and Andy Farnell for intelligent comments - and for providing an informed sounding board for my ideas. To David Knight and Orville Jones for granting me the opportunity to embark on my research, and again to Orville Jones for getting me into software engineering in the first place.

Thanks to all those at the collaborating establishment who participated in the study of process. Notably Dave Symes for his vision, numerous ideas, help with organization, and his general backing and support throughout the on-site work, Herman Desmier for supporting the investigation of the launch process, the instance model project mangers, and especially Howard Whalley, and Antony Spilman for taking time not just to co-operate in the investigation but to make sure that I was getting the information that I needed, and that it was as accurate as they could make it.

Thanks to Peter Henderson for his patience and understanding.

Thanks to Chris McManus, Joel Chapman and Stuart Nicholson for helping me to keep my sanity, and to my parents for their continued support, sympathy and understanding.

Finally thanks to Michelle, for getting me through the last lap of the write-up, and for putting up with me.

Contents

1. Statement of the Problem

Chapter Synopsis

This chapter introduces the topic of software process modelling, and defines the aims, scope and context of our research. We include an overview of our work and pointers to more in depth discussion in later chapters.

1.1 Background

1.1.1 Difficulties with Software Development

Software development is notoriously troublesome. The fabled 'software crisis' seems never to have really abated, and cost and schedule overruns, unsatisfactory products, and even products which never become used are far from rare even today. These difficulties are made more apparent by the astonishing gains in hardware over the same period; as Brooks states 'one must observe that the anomaly is not that software progress is so slow, but that computer hardware progress is so fast. No other technology since civilization began has seen six orders of magnitude in performance-price gain in 30 years' [1]. Brooks believes that the nature of software means that there will be 'no inventions that will do for software productivity, reliability and simplicity what electronics, transistors, and large scale integration did for computer hardware'.

This belief that it is the intrinsic nature of software, and hence the intrinsic difficulty of software development which is to blame for the failure of software projects is shared by others. Kitchenham notes that: 'In other industries it is usual to produce the same types of product over and over again. In the software industry it is usual to produce new products using different methods and tools' [2]. This problem of the 'one-off' illustrates that software development is really an exercise in design, with no real production process as such. Consequently the manufacturing paradigm, which has been successfully applied to other engineering disciplines does not map easily to software development [3, 4].

The development process is also a learning process, in that many of its participants are actually learning both application domain and technical knowledge throughout the development of the product [5-9]. This again, is quite rare, for in most other engineering disciplines application domain knowledge is of a very high level, before a project is started. Thus, managing the process of software development is actually managing a communication and learning process where there is a high proportion of design activity. Rodden et al. sum up the consequences of these problems by stating [10]:

'Software development is complex and time consuming. It involves a considerable investment of people, resources and time. Severe problems exist in understanding the activities involved in software development and in effective management of the process'.

It is due to these difficulties with the process of developing software, and with understanding and managing this process that software project failures are still so

common. Software projects which are over budget, over schedule or which produce software which does not satisfy the customer are common-place[1].

1.1.2 Panaceas and Process Models

Having presented some of the difficulties with software development, it is now worth considering ways to alleviate these problems. One technique that has been proposed is that of software process modelling. For now process modelling can be thought of simply as a way of:

capturing** (or understanding) **and describing the software process, for some purpose.

Process modelling promises benefits in a number of areas essential to software engineering. Process models can play a role in the comprehension, design, support, integration (of methods, tools and activities), evolution and reasoning about process [12]. They can improve the development process itself, by identifying effective process activities, and by increasing the co-ordination of those process activities. Furthermore, by increasing the effectiveness of the process activities, process modelling can help to improve the quality of software products produced.

However, many authors have cautioned against searching for a panacea, a cure-all for software development - the most well known argument being Brooks' statement that there is 'No Silver Bullet' [1]. The software engineering community has already suffered from too many past 'panaceas', and we must be careful not to consider process modelling is this way. Rather than arguing that modelling alone is the answer, many believe that one of the important roles that process modelling can play is in providing an integrating framework for other initiatives like CASE, or software measurement [13-18]. Indeed, it may even be argued that it is the lack of such a framework which has been responsible for the failure of many past initiatives. It is the potential of the technique, the contribution that process modelling can make to understanding and improving the software process and product, that makes it such a worthy candidate for research, and which explains the interest which it has created within the software engineering community [19-29]. Process models may make the software process more visible, and understandable; analysis of such models may aid the identification of problems or process areas which may be improved, and models can be used to facilitate communication about the process, and experimenting with new process design. Finally, as we noted above, process modelling may provide a way to integrate a number of apparently disparate technologies, including CASE tools, and software measurement. (A fuller description of the potential use of process models is given in the following chapter).

1.2 Aims of this research

Process modelling is a potentially powerful technology, which has stimulated a great deal of academic work over the past ten years. However, there is relatively

---

[1] Curtis notes a US Airforce survey which found that all major projects in one command were late, and that these late projects were on average 75% over schedule [11].

little reported experience of the use of this technology within industry[2]. Thus, we have the first aim of our work:

1) To investigate process modelling in practice.

However, we also wish to use our experiences of modelling to identify ways in which we can aid the technology transfer by making process modelling both accessible and effective. Thus, our second aim is:

2) To make process modelling more effective.

We now examine these ideas in more detail in the following sections.

1.2.1 Process Modelling in Practice

This aim will be the focus of our first more exploratory study. Our objectives for this study are to investigate the use of process modelling to aid understanding and to provide insights about software development. Hence, we intend to use simple process models to help us understand some aspect of software development at an industrial site. It is important to note that we are not attempting to demonstrate process improvement, but the utility of process modelling. Hence, the study is concerned with getting process users to buy-in to the modelling, so that the model can be used to improve their understanding of their processes. Therefore, we intend to use process modelling to help to understand or solve an existing and genuine process problem. Thus, our aims are to:

1) Investigate the types of modelling techniques available.

2) Investigate the implementation of a defined process at an industrial site.

3) Investigate whether a simple process modelling technique can highlight problems with the implementation of the defined process.

We believe that the benefits of such work will include:

1) Process modelling will be perceived as a technique which can solve genuine problems.

2) Evidence of the successes of such work will encourage others to carry out process modelling work.

3) The experience gained from such work in industry will be valuable if we are to provide guidance to other practitioners.

1.2.2 Making Process Modelling More Effective

In the course of our exploratory modelling we expect to uncover certain insights about the process under scrutiny. By using this as an example process we hope to investigate what further insights can be gained by using new modelling notations or strategies. More specifically we hope to:

---

[2] This lack of reported industrial experience was one of the main themes of the Third European Workshop on Software Process Technology EWSPT'94

1) Enhance the process modelling techniques in order to make process problems more visible (and to uncover further insights into the process under scrutiny).

2) Test the new technique to see if it can distinguish the differences among actual projects at the site.

In addition, we hope to be able to:

3) Show how process modelling can help us to identify good practice, or effective process activities.

4) Investigate the use of process models for providing a framework for other techniques, specifically software measurement.

1.3 Scope of the Research

Process modelling is being used in a variety of disciplines and for different purposes. In order to make the scope of our work clear to the reader, we briefly describe these related fields below, contrasting them with our own work. However, for those who are familiar with such disciplines we need only distinguish our work by noting that:

1) We are not using process modelling as part of an effort devoted to building a process support environment or an integrated project support environment.

2) We are not attempting to use process modelling as part of a business process re-engineering initiative.

3) We are specifically interested in examination of the software development process, not in business process modelling in general.

However, the part of the software development process which we are investigating could be regarded as a business process - or part of the larger business process. In addition, we believe that the techniques which we are using could equally well be applied to the business process. The main distinction is that we are not re-engineering, because we are specifically adopting a far more evolutionary approach (see section 1.3.2).

1.3.1 Business Process Modelling

Process modelling is gaining increasing popularity within the business community, as a way to describe and improve business processes. Indeed, much work which was initially aimed at describing and improving the software development process, has been taken up by the business process improvement community [30-38]. For example, both Praxis [35] (now part of Deloitte-Touche-Tohmatsu) and Co-Ordination Systems [33] are using Role-Activity Diagrams as a business process re-engineering technique and VSF are marketing their Business Improvement Facility (BIF) - which is based on the VSF case tool - as a business modelling solution [37-39]. The significant overlap between these disciplines (business process modelling and software process modelling) has meant that initiatives like IOPT (Introduction of Process Technology scheme which was initially sponsored by the DTI) have included many from both camps, and there are similar tools and techniques being used in the quest to improve both business and software

processes. Indeed, both ICL and Cap-Gemini have tools - ProcessWise (ICL) and Process Weaver (Cap-Gemini) - which appear to be marketed both at the software process (as software engineering environments) and at the business process communities [40-42].

The aim of our research is to investigate software process modelling, and not business process improvement. This means that we examine processes that are explicitly used to produce a software product, and which can be considered as vital to software development. However, the impact of business process improvement is significant, notably in providing a market for tools which were initially developed for supporting software development projects, and in helping to raise the wider awareness of process modelling. Furthermore, there is no significant difference in the notations used by software process modelling and business process modelling. For this reason, it has been suggested that the business and software process modelling communities should attempt to work more closely, and to learn from each other [43].

1.3.2 Business Process Re-engineering (BPR)

BPR differs from other business improvement programmes, in that it is explicitly a revolutionary rather than evolutionary approach. Mc-Hugh who was involved in the evolution of the Deloitte-Touche-Tohmatsu / Praxis method STRIM (Strategic Techniques for Role and Interaction Modelling) has referred to it as a 'quantum leap approach' [36]. Kawalek and White make a further distinction between: 'process engineering'- a continuous activity, and business process re-engineering - typically a one off [40]. In addition, BPR is more concerned with designing new processes than it is with examining existing ones. Hammer and Champy [44] state that in order to succeed in re-engineering: 'Don't focus on business processes'. They further state the case by arguing that we should 'Ignore everything except process redesign'.

BPR attempts to avoid focusing on what is considered implementation detail. For example, Miers [45] warns that in the early stages of BPR ' When modelling the existing business process the use of existing data and documents should be specifically excluded from the model ...'. This is to make it easier to break down links with the existing process, when attempting design, or radical re-design of the process. This is clearly in contrast to the evolutionary approach of business and process modelling (and business and process improvement). Thus, while our own study involves investigation of a business process it is not business process re-engineering (BPR).

1.3.3 Workflow

The motivation for using workflow software is hinted at by a quotation from Dwyer (director of strategic marketing at XSoft) who has stated: 'The reason cited most often for installing workflow software is that people want to re-engineer a business process' [46]. However, current workflow technology, is very much orientated towards replacing documents and paper based systems and Dwyer notes that users are "typically an organization which needs to handle a high number of paper-based transactions each day, such as insurance companies or mortgage lenders". This suggests that workflow technology is most suited to providing support where there is a high number of relatively well understood tasks, and though it has been suggested that future workflow software might also be used in the production of software there is little evidence that current workflow systems have the sophistication to do this [46].

1.3.4 Software Process Modelling

We have suggested that there are many similarities between business process modelling and software process modelling. However, there appear to be important differences, not in the underlying rationale but in the way in which the technologies have been used. On the whole, business process modelling has been used in a revolutionary way, as suggested by BPR, which usually involves complete process re-design or replacement. Though business process modelling may still involve an investigation of the original process (though not always) the intention is usually to change radically the way people work rather than to 'tweak' or improve the process. Software process modelling appears to be seen as a more evolutionary approach. This may be to do with the highly complex nature of software development, or indeed the difficulty of designing and implementing a 'whole new process', but it is far more likely that the software process modeller is aiming at process improvement rather than more radical process change. This difference in approach also leads to differences in scale, in that the software process modelling may be directed at improving part of a larger process (as in our work where we examine part of the development process), rather than suggesting a re-organization of the whole business or organization.

1.3.5 Process Support Tools and Environments

Much work on software process modelling has been prompted by the desire to build software project support environments. This work has been motivated by the need to first understand the development process, and the way in which people work, in order to be able to provide appropriate automated support. Such work is being carried out by a number of organizations, and there are already many existing tools. However, the cost, and complexity of many of these tools makes them inappropriate for many smaller developers.

Our research is not concerned with either building or using such environments, nor is it an attempt to simulate or automate the process. There are two main reasons for this. First, the aims of the research do not require such elaborate tools. If we can demonstrate the efficacy of process modelling without recourse to such complex and expensive tools then this research will have been successful. Secondly this research intends to investigate process modelling techniques in an industrial environment. For the organization that has collaborated in this research (which is one of the many smaller developers) the considerable investment it would incur is simply not feasible. In addition, the strategy adopted is to start with a low-cost study and demonstrate the benefits of this work before attempting a more ambitious study, and this strategy prohibits such heavy up-front investment.

This low-cost approach has been somewhat out of favour recently, however, there are signs that this view is changing. For example, Ince predicts that 'notations used for software process modelling will be no more sophisticated than the simpler CASE notations that are currently used for requirements specifications, such as dataflow diagrams' [34].

The need for reported experience of software process modelling in industrial environments was one of the strongest workshop themes at the recent European Workshop on Process Support Technology (EWSPT'94) [29]. It is the belief of this research, that the low technology or low cost approach is a necessary way to provide not only experience with but also evidence of process modelling successes. The lessons learned from such 'real' work will far outweigh its theoretical limitations.

1.4 Context of the Study

This process modelling study was carried out in order to attempt to understand why there were problems with an existing software development process, and to discover what elements of that process had an impact on project success. Hence, the study was not concerned with demonstrating process improvement, or with radical process design or re-engineering. We were motivated to understand the current process, and its problems, in order to evolve that process.

In the following section (1.5) we outline the work which was undertaken (which will be further described in chapters Four to Six). However, we first describe in greater detail the setting (site) for the study and the specific problem that the work addresses.

1.4.1 The Collaborating Organization

Work was carried out at one site which manufactures products with high software content and complexity, and where the distinctions between hardware and software engineers are often blurred. There are about 50 full time software engineers, though projects include input from other divisions at the site, notably from hardware, marketing, and production.

The main business area of the organization which we investigated was the engineering function, which encompassed both hardware and software design, though the projects that we studied were those where the majority (or all) of the work could be considered software development. However, owing to the nature of the process area under study we also examined other business areas which affected the process, notably marketing and the separate project support function. Projects at the site varied in both size and complexity, but usually involved between six to ten engineers. A project manager would be responsible for driving each project, but the project team would typically involve personnel who reported to other line managers. Often projects formed part of a larger (encompassing) project which could mean the involvement of many more (for example, around thirty) people.

There were no set software analysis or design methods, and no standard CASE tools employed at the site. The majority of coding was of a procedural nature (C being used extensively), and design documents tended to reflect this. Similarly engineers and managers alike, seemed happy with procedural notations, and tended to think of the process either in terms of activities or products.

Textual procedures documents were of a number of forms, some were general to the site, for example those detailing the development process, some were specific to business functions (for example engineering procedures) and some were even more localized (e.g. hardware or software design procedures). There was a genuine attempt to have 'quality' procedures and process, and the process quality manager was recognized as an important (and senior) role.

1.4.1 The Process to be Investigated

The process which was chosen for investigation was termed product or project 'launch'. This 'launch phase' of software development covered the stage from the identification of a project or product need, through business and technical feasibility stages to requirements. These requirements were intended to be sufficient for the

commencement of design work[3]. This area of process had a troubled history, with three recent incarnations - or process designs. There was also an awareness that the existing process was not adhered to as its designers had envisaged. Both users of the process and designers of the process expressed dissatisfaction. Indeed, this process area had historically proved overly time-consuming, difficult to control, and a cause of much disagreement. Consequently a great deal of effort had been expended on the current process design but there was still much dissatisfaction.

Though the process designers could not understand why the process was still problematic, they knew that they were unable to completely distance themselves, and that problems might not be the fault of process users. It became clear from our discussions that the organization might benefit from an independent view - even if only to confirm their suspicions that something was amiss.

1.5 Overview of the work

1.5.1 Exploratory Work

Our exploratory work had the following research aims.

1) To discover the realities of process modelling in industry.

   How do you go about process modelling? What are the problems or pitfalls?

   To use our experiences to provide lessons learned for the modelling community and for other actual or potential practitioners.

2) To investigate whether process modelling can aid process understanding.

   We wanted to know if modelling could provide insight about the area of the software development process being modelled, and if so what kinds of insights we gained from different techniques.

   For example, we wished to investigate the extent of process conformance at the site. Was there a standard process, and how did users deviate from this process? Could we use process modelling to highlight and to understand this process deviation?

3) To show that simple (existing) diagramming techniques can be used for process modelling.

   Furthermore, we wanted to show that a single paradigm can be used to produce process models which still provide useful and usable results.

4) To identify areas of potential process modelling improvement.

From the point of view of our collaborators, the aim of the work was to use process modelling to uncover and describe process problems. In order to do this we undertook preliminary modelling utilizing data flow techniques to compare and contrast the actual process (for projects) with the theoretical process (as depicted by

---

[3] Note that this life-cycle view of where the process area fit was one which was consistent with the procedures, rather than imposed by us, but it does help to reflect the activities and products which were expected from this part of the process.

procedures and supporting documentation). This initial study yielded useful results, and we were able to discover and describe half a dozen key process problems, and many minor discrepancies and inconsistencies.

1.5.2 Later Work

The research aims for our later work were as follows.

1) To devise and use new notations or techniques to provide further insights into the launch process. Hence, to show how process modelling could be made more effective.

   For example, to further investigate the extent of process conformance by examining differences among projects which should follow the same standard process.

   Another example which was of particular interest to the organization was to use process modelling to try to identify key aspects of the launch process, and their effects upon the projects.

2) To investigate the use of process models as a framework for data collection.

   To show that we could combine process measures with our models so that the process models can be used as a framework for data display.

Of the above aims the organization's chief concern was in using this process modelling study to try to ascertain the impact of launch process activities on project success. In order to uncover the impact of process activities we needed to do more than simply note that such an activity occurred. Rather we conjectured that the amount of effort expended on an activity would be a factor in its impact on project success. For example, we would expect that spending eight hours planning a project might have a different impact to spending eight minutes. Ascertaining which activities had this kind of impact would enable the organization to provide strong process guidance about what was worth spending time on, and similarly what corners could be cut. Consequently we decided to incorporate data collection into our process modelling study, both to determine information about launch activities and to examine project success. Rather than simply have activity based models with associated data, we have developed a notation to show the effort and duration of each activity in pictorial form. Each activity is represented by a rectangle, scaled so that its horizontal axis corresponds to its duration and its vertical axis to the average effort over that duration. Hence, the area of the activity represents the total effort expended. Activities are then shown against an overall project time-scale, so that we can see the overall shape of the project. This TRADE (Time Resource Activity Duration Effort) notation has allowed us to uncover information about which activities have positive impacts upon project success. However, perhaps more importantly this notation and study has allowed us to uncover the nature and extent of deviation among projects at this site. Hence, we believe that we have satisfied the above research aims.

1.6 Summary

We have noted that the continuing difficulties with software development have meant that there is still a search for methods and techniques to help us understand, manage or control the software process. Software process modelling is an approach which offers much promise, but which has still received relatively little industrial

use. We have further described our research aims, which centre around the need to investigate process modelling in practice, and have outlined the scope and intent of this work. We have also described the background to our work. We have given a description of the context (the site) and the process problem, and we have given an outline of the work which has been carried out. The following chapter describes in some detail the history and issues of software process modelling. We argue why there is a need for work such as ours, and attempt to distinguish this work from that of others.

Later chapters then describe our rationale and choice of research methods, the exploratory and later studies, the results of our work, and finally our analysis, findings and conclusions.

2. Literature Review

Chapter Synopsis

This chapter starts by considering the promise of software process modelling. Following sections examine the significant events or ideas that led to current thinking about process modelling, some related areas of work, the current state of the art and the process modelling issues remaining to be resolved. Finally we set our own work within this context, and show both the need for our work, and how it differs from that of others.

2.1 The promise of software process modelling

We have suggested that process modelling has much potential in software engineering. We now list some of the most important uses which have been suggested for software process modelling.

1) Models should provide higher process visibility which will in turn lead to a better understanding of development processes [47, 48], aid the monitoring of progress, and allow managers to give better guidance to engineers [12, 49, 50].

2) Explicit descriptions of such processes should encourage better communication about the process [50].

3) Through analyzing the model of our process we should be able to more easily identify areas of weakness and possible improvements [49]

4) By providing a framework for software measurement, quantitative data about either process or product may be gathered more efficiently [17, 18, 51-53].

5) Models should allow us to experiment with process [54]. This experimentation could be done at a purely conceptual level, simply by using the model to more clearly express ideas and reason about the process. In contrast automating the model will allow us to simulate a process, step through the logic of that process, and experiment with the effects of process change. Thus, models will support process evolution [48, 50].

6) We can use a model as a process template that can be instantiated for each project [34, 55]. This template gives us the benefits of standardization while still allowing process flexibility for individual projects [34, 56].

8) Models should facilitate process reuse [14, 57, 58]. This reuse may occur at a number of levels: for example, instantiating the process for each project, using the model to provide guidance on the recommended process, or using the model as a repository of process knowledge.

9) A number of quality initiatives suggest the need for a defined process [59-62]. Process models can form part (or all) of this definition, and can thus give the organization tangible benefits (such as accreditation) as well as

those (often more long term) process benefits mentioned in the points above.

2.2 Influential Work

2.2.1 Lifecycle Models

Early attempts at describing the software development process can be classified as 'phase' or 'lifecycle' models, in that they tended to attempt to split development activity into chunks (phases) of different activity types. For example, requirements elicitation, specification definition, design, coding, and so on are regarded as distinct sequential tasks each being performed only when its predecessor is complete. Thus, the entire lifecycle can be decomposed into these discrete activity areas. The earliest of these, the nine-phase model of software development, was proposed by Bennington in 1956 [63] and is now better known as Royce's waterfall model (1970) [64]. Such models remained popular for some time and are still described in many current student texts [65, 66]. An advantage of studying these models is that, despite their limitations, they still convey the various types of software engineering activity to the uninitiated. In addition, the end of each phase can be used as a rather crude milestone against which progress can be judged. However, there are a number of problems with lifecycle models, and one of the early workshops questioned the ability of the phased or lifecycle models to describe actual software development. We briefly mention three key problems with lifecycle models:

1) Iteration

   Such models do not adequately reflect iteration. In reality, requirements activities do not end at the beginning of design, or even coding. Rather they are changed, deleted, updated and appended. Similarly design, code, documentation and test suites are changed as understanding of the problem grows. Iteration is a problem of such consequence that the Third Software Process Workshop addressed explicitly 'Iteration in the Software Process' [24].

2) Prototyping

   The need for prototyping significantly altered the concept of software lifecycle [67, 68]. For example, the assumption that development must follow a rigid sequence from specification through design to coding was challenged [69].

   Boehm's 'Spiral model' incorporates both iteration and prototyping, by cycling through the various development activities and pausing to assess risk and change [70-73]. Though the spiral model provides only a large-grain view, the rejection of the rigid phased approach and the inclusion of prototyping depicts a significant change in software development is organization.

3) Granularity

   A further problem with lifecycle models is their large-grained view. If the level of abstraction is rather high, what is suitable as an overview cannot really represent the complex interactions occurring at a much lower level. Hence, Madhavji claims that these models, though helpful, 'do not expose myriad details that are critical in any large software development project.' [74].

Owing to these problems the idea of modelling the development process with lifecycle models has been largely rejected. However, much of the terminology that introduced by the models (e.g. requirements phase, design phase and so on) still remains.

2.2.2 Process Programming

Process programming, typified by the work of Osterweil [75, 76][4], regards the development process as a set of activities (and their associated inputs and outputs) that can be described by a procedural language, in the same way that a software program describes the data and flows to be captured in the software.

This view has five significant implications.

1) There is a software process lifecycle.

   Osterweil's analogy that 'software processes are software too' [75] is perhaps best illustrated by his statement that 'the various software processes should be viewed as having been created by process development processes'. One of the implications of such a statement is that software processes have a development lifecycle in much the same way as software products do. Rather than a normal product lifecycle, the process lifecycle is more a description of how process models might evolve, serving as a guide for introducing models or as a framework for planning such work. This argument was made explicit by Kaiser [86] who states that the analogy was originally suggested by Boehm and Belz [71]. A brief description of the phases of this lifecycle is as follows:

Process Requirements:	Deciding what we want to express with the process model.
Process Design:	Devising a means within the chosen formalism for meeting the requirements.
Process Construction:	Writing the model or program within some formalism.
Process Testing /Debugging:	Testing the process correctness using simulation, activity assistants [87] and actual projects.
Process Evolution:	Evolving from one process model to another.
Process Re-Use:	Reusing part or all of the process model.

   Although developing and evolving software process models according to this kind of specific lifecycle have seen little work of late, the notion of a method for constructing and evolving process models can be seen in current work on the meta-process [88-90].

2) Modelling languages should be like programming languages.

   Osterweil argues the need for both products and processes to be 'carefully and rigourously specified in terms of a rigourously defined language'. Thus, we see models of the development process that use programming-like constructs and look very much like software programs [78, 91, 92]. The

---

[4] Other examples of process programming work include [77-85]

advantage of such models is that they cope with the kind of low-level detail which is not covered by lifecycle models. However, a weakness is the conviction that 'all software activities must be viewed as being aimed at the creation and/or alteration of software products' [75]. Hence, other supporting activities, which may be an important part of the development process may be ignored by the resulting model. For example, Curtis explains that communications tools, though they aid the development, will not transform the artifact and would thus be ignored by the process programming approach [8].

There is now increasing agreement that we cannot describe formally all aspects of software development. We cannot model creative human processes (see below). Thus, we must consider what parts of the development process we should attempt to formalize [93]. However, the need to make modelling languages accessible to users is the biggest argument against program-like notations. Process programming notations, are often highly mathematical [80] or program like [92] and their use requires existing expertise or learning. Many of the uses of process models involve domain experts or customers who are not software or mathematics experts, validating the model, discussion about process, using the model to communicate ideas, getting process buy-in, and so on. In these cases formal notations may obscure information or hamper communication.

3) Human behaviour can be codified.

One of the problems with trying to codify human behaviour is that it assumes that we completely understand that behaviour and that the code will be able to describe all relevant aspects of it. This may be the case when humans behave in a mechanistic way, for example, carrying out a series of simple and well described tasks. However, much of human behaviour is not like this. In particular learning does not follow such mechanistic paths. Indeed, learning curves are far from linear, and the mapping between mental models and logical or physical models not clearly understood. Furthermore, it is extremely difficult to provide complete instructions to enable humans to satisfactorily carry out complex tasks. Few people would be able to drive a car safely by simply digesting a driving manual. Additional human guidance, though often less rigourously stated, is required.

Much of software development involves understanding and learning. Henderson suggests that this may involve as much as 50% of developers time [43]. Therefore, it is unlikely that the software development process can be completely codified.

4) The process may be automated.

Program-like models can be easily executed. Thus, process programs often lead to executable models[5]. Such models could be used for simulation or for automating part of the process. However, the process program or the enactable notation increasingly forms only part of an approach to process modelling. It is now far more common for process support tools to present a graphical language to users, which may then map to some executable notation (e.g. [33, 41, 96]).

---

[5] Note that executable models, and automated process support had of course been considered before the introduction of the process programming analogy [94, 95].

5) We can rigidly control the software process.

Degree of control has been a point of much debate in the process modelling community. Some of the strongest arguments against attempting to use process programs to control the process have been voiced by Lehman[6]. Lehman feels that there is a temptation to have too much detail with process programs, and that they may act like a 'straight-jacket'. His overall prognosis is that process programs are powerful for analysis, design and so on, but that control depends upon people and context [98].

We have noted many arguments here about the appropriateness of process programming, citing debates about:

1) Whether human activities can be formalized or codified.

2) What it is appropriate to formalize.

3) Whether we can rigidly control the software process.

4) Whether process programs are sufficiently usable as a modelling approach.

We believe that process programming is losing favour, because there is greater agreement that people cannot be rigidly controlled, because we cannot completely codify human behaviour and learning and because process programming notations are not accessible or usable for many developers. However, despite much debate about process programming, there has been little attempt to validate the claims (of either proponents or opponents) by empirical study, or by observation of process programming in industrial practice.

2.2.3 Integrated Project Support Environments

A great deal of process modelling work has been motivated by the wish to build project support environments and the desire to automate the development process. However, such work, particularly the earlier work on integrated project support environments (IPSEs) [95, 100-111] has had limited success [112]. From a modelling point of view, one of the problems with many of these early environments was that the model of development or models on which they were based were often questionable. Though some researchers attempted to understand the process of development (for example ISTAR was based on an explicit contractual model [113-115]), others were more concerned with providing a way to bolt together tools, and they made little attempt to understand the way users actually worked [110]. To compound this problem project support environments were often based upon an implicit rather than an explicit model of the software process.

---

[6] Lehman (like Balzer [97]) believes that there are fundamental differences between process programs and application programs [98]. He concedes that the apparent advantage of a process program is that a more formal description can be machine interpreted, and thus used as a control mechanism. However, he considers that such a program is little more than "a series of calls for specific actions", and that it still needs human intervention and decision making. It is this human intervention which is still not well understood. Lehman argues that "detailed process structure and composition cannot be determined", and that "process descriptions whether formal or informal, are essentially imprecise and non-deterministic." [99]. A further inherent problem which Lehman considers is that whereas the application domain is generally continuous and infinite, computer based process models are discrete and finite. Therefore **program-based process models actually limit 'the scope and power of what can be achieved'.**

Humphrey states that 'To qualify as a process model, the process used by the tool should be explicitly defined' [54]. Despite much research work [116], these integrated project support environments (IPSEs) never lived up to expectations [95, 109-111].

It is, therefore, interesting to note that although we now hear very little about integrated project support environments (IPSEs), many of the process support tools now in vogue share language and notational features with these IPSEs, as well as claiming similar benefits. For example, ICL's ProcessWise Integrator (PWI) relies on the same modelling language (PML) developed as part of IPSE 2.5 [117].

2.3 Related Work

2.3.1 Process Support Tools

We have noted that much of the work on providing automated process support has switched emphasis away from the concept of an integrated project support environment (IPSE) towards that of providing process support technology. Despite this, other researchers are still attempting to build software engineering environments [86, 118-125]. This situation raises a number of questions:

1) How are current attempts at building Software Engineering Environments different from those involving early IPSEs?

   There now appears to be a much stronger focus on understanding and supporting the way humans work, rather than simply producing a suite of tools, and on making the model of the process explicit. For example, Process Weaver places great emphasis on being process-centred. SPADE [118, 126] explicitly separates the process model from the process engine (which enacts that model), the process tools, the user interface, and the environment itself, such that these elements can all be independent[7]. The SPADE approach allows not only greater process model visibility but also greater flexibility about the modelling languages used and the interfaces provided, according to needs of users.

2) How are process support tools different from environments?

   One might expect that we could use the early IPSE intention of integrating all aspects of development to distinguish environments from process support tools (or suites of such process tools) which offer process support for specific parts of the development process. For example, the ICL's ProcessWise [128]) consists of a number of tools that tackle different development tasks. However, the intention is clearly to have these tools in some integrating framework, and this appears to be very similar to the IPSE concept. In some cases, one might wonder whether there is much more than a difference in terminology, in that what are termed process support tools appear to offer similar capabilities to environments, and are often based on the same earlier IPSE work (e.g. IPSE 2.5 [129] and ProcessWise [128]).

   One clue to the shifting terminology is that much work has moved away from a concentration on software development towards supporting business process improvement and re-engineering [33, 37-39]. An example is the

---

[7] Note that the question of how dependent the architecture is on the Process Modelling Language is still very much an issue in the research community [127].

way VSF have shifted emphasis away from the meta-CASE aspect of their tool towards selling business process modelling solutions[8] [37]. One might consider this a re-badging of an established technology in order to make it more palatable to another market sector (more of a change in emphasis and marketing, than in underlying tools).

An advantage of this change in terminology is that it allows the process support tools to be introduced to a software engineering community who, remembering the failures of early software engineering environments, may be reluctant to purchase another IPSE.

3) How does this fit with the software factory concept?

The ultimate in automating development is the idea of the software factory [130]. Indeed, Longchamp considers the software factory to be the third generation of the integrated project support environment [131][9], and considerable effort has been devoted to producing automated support for the factory concept[10] [116, 134, 135]. It is interesting to note that as with the IPSE work mentioned above, this third generation work has resulted in tools that are now marketed under the process support technology banner (e.g. Process Weaver).

There are currently a number of impressive process modelling tools available (which may be variously labelled as process support tools, process technology tools or as software engineering environments). There are clearly many similarities among these tools; all are attempting to automate the process (business or software) to a lesser or greater extent, and all rely on producing or enacting process models. Rather than a clearly defined difference between these classes of tools, we view this as a shift in emphasis, and thus we have chosen to consider them all under the heading of process support.

These process support tools typically offer both flexible graphical interfaces[11] and process automation engines to provide enactable models. We take issue not with the capability of these tools but rather with their usage. Despite the fact that these tools have enaction capabilities, many have reported that when tools are being used in industry, they are only being used for the initial process modelling, and not to provide process automation and enaction. For example, Griffiths notes that people are using ProcessWise to produce "soft wall charts" [140]. We believe that this

---

[8] VSF have developed (in conjunction with N&P) a proprietary modelling method, Business Improvement Facility (BIF), to be used with their tool.

[9] Much disagreement has centred on the applicability of a factory or manufacturing paradigm to the software process [3, 132]. Proponents of the factory concept argue that despite the fact that there are significant differences between producing hardware and software the similarities outweigh the differences, and that we can usefully apply the paradigm. [133].

[10] Gillies notes that the term software factory "has been adopted by the large European ESF research program aimed at producing a state-of-the art software development environment (SDE)...."

[11] Typically the graphical language maps onto some executable notation to provide the enaction capability. One problem with this approach is that it may limit either the expressive power of the graphical notation or lead to an inconsistent or incomplete mapping. An example of this is the mapping between Role Activity diagrams (RADs) [35, 136], a graphical process description, and Process Modelling Language (PML) [35, 137-139]. These are intended to give a high level and a more detailed view of process, respectively and the RADs should map to PML. However, the RADs contain specific mechanisms to describe parallelism, whereas the PML being constrained by its need for compilation does not, and thus the mapping is incomplete.

indicates the inappropriateness of these tools for a large number of development organizations. The fact that many are using such impressive tools simply to understand or describe their processes rather than to automate them suggests that such enaction capabilities are merely 'bells and whistles' which tool builders may be keen to incorporate but which users either do not need or do not wish to use. Indeed, this rejection of enaction, even when given the capability, suggests that a more low-technology approach may be more appropriate.

2.3.2 Process Assessment and Capability Evaluation

An important and related area of work is process maturity assessment and capability evaluation. There are now a number of process assessment and evaluation frameworks [59, 61, 141-146], but perhaps the most influential work in this area has been that of the Software Engineering Institute (at Carnegie Mellon) on Software Process Maturity [147, 148] and the Capability Maturity Model [60, 149]. This model was derived from work at IBM [150] under the direction of Humphrey, which attempted to apply Crosby's quality management grid [151] to the software process. The Capability Maturity Model (CMM) gives five levels of process maturity. It is intended to provide guidance for process improvement by focusing on key practices and activities within the organization. Rather than modelling the existing software process, the method involves bench-marking against a defined list of acceptable or desirable criteria using a questionnaire. These bench-marks have been obtained by surveys of best practices, tools and methods.

The idea has been adopted for two main, though not entirely distinct, purposes:

1) Assessment of the capability of an organization.

This idea was particularly attractive to the US Department of Defence, who wished to be able to assess the capability of contractors. Indeed, this was one of the motivations for the work by the SEI. An assessment by the Software Engineering Institute or an approved assessor leads the organization to be categorized by maturity level. The least mature is level one at which the process is said to be ad-hoc. The most mature is level five, where the process can be changed dynamically as improvements and data are fed back in.

LEVEL	CHARACTERISTIC
5. Optimizing	Improvement fed back in to process
4. Managed	Measured Process (Quantitative)
3. Defined	Process defined and institutionalized (Qualitative)
2. Repeatable	Process dependent on individuals
1. Initial	Ad hoc or chaotic process

2) To aid process improvement

Curtis describes each maturity level as 'a well defined evolutionary plateau on the road to becoming an exceptional software organization' [152]. The intention is that the model helps the organization to focus on appropriate areas for improvement depending on the maturity of its current process.

Later versions [153] of the maturity model aid this approach by identifying key process areas and key practices for each level.

This idea has also been combined with other software engineering initiatives, notably software metrics, by using maturity assessment in order to determine appropriate actions. For example, the application of metrics in industry (ami) handbook [154, 155] suggests using maturity to help characterize an organization as the first step in a metrics programme. More specifically, Pfleeger and McGowan use each maturity level to suggest sets of metrics which are appropriate for the organization to collect [156], and in further work at Contel, Pfleeger suggests using maturity [157] to aid in CASE tool selection.

There have been a number of criticisms of the SEI approach. Some of these have focused on what may be inherent problems with the method [158], whereas other researchers mainly question its applicability to all types of software development organization [159-162][12]. We will briefly examine some of these criticisms.

1) The problem with using a bench-marking based on best practices is that it is not clear that there were any excellent organizations on which to base the higher maturity levels. Certainly there were no level 4 or 5. Therefore, these top two levels are based only on inference [158].

2) Improvement is self-fulfilling, and no other independent evidence of process improvement is given [158].

   If the organization does comply with the practices suggested, then it will by definition improve its process maturity score. (This is irrespective of whether it produces a more successful product, for example). Furthermore, there is no empirical evidence that the 'good practices', are actually good and effective.

3) The assessment focuses only on good practices, and has no focus on eliminating bad practices. That is, it is only 'half the story' [158].

4) The ordinal maturity scale bands together the majority of developers even though they may have quite different real capabilities. [162-164].

   Studies to assess the current maturity level of organizations in Europe and the United States suggest that over seventy percent are at level one [164]. Among this 'majority' of software developers there are huge differences [161] and yet because the scale is ordinal rather than interval there is no way of distinguishing between those who are 'nearly level two', and those who are 'nowhere near'.

5) There is too much focus on a maturity level or score and not enough on improving process [144, 163].

---

12 Thompson feels that the maturity model must be tailored in order to make it appropriate to Information Systems (IS) developers, and that "...a number of key IS development activities are not present in any levels of the model...". However, the problems he notes mainly focus on key differences between defence orientated and information systems developers, specifically that :

a) IS organizations have a different internal culture.
b) IS developers have a different external environment
c) IS developers are involved in a different type of software development project.
d) IS developers have a different development approach.

The concentration on doing the 'right things' to improve one maturity score is not necessarily the same as doing the 'best things' to improve the process. Koch rejects the idea of an absolute ordinal maturity scale in favour of 'improving software processes by self-referential improvement exercises'. He describes this as the 'Central Concept', behind BOOTSTRAP (a European Software Capability Assessment Initiative) and notes the influence of the Japanese concept of 'Kaizen', which he compares to the western paradigm of 'Radical Constructivism (the paradigm of the self)'. In addition, the Capability Maturity Model provides no guidance on which key practices should be implemented first for a given level, and no concept of the relative cost benefits of those key practices.

Many of these criticisms, are being addressed by the SEI, and it would be wrong to view the maturity model as static and monolithic. For example, the version 1.1 [153] emphasises key activity profiles rather than absolute scores [165]. However, it is important to realize that there is no empirical evidence for the success of the CMM. We have no way of knowing that it is the effect of the CMM that makes a difference to the software developer. For example, the assessment practice involves a significant amount of resource being devoted to examination of current processes. It may be that it is the much needed examination of current process which yields change and improvement and that the CMM is merely a vehicle for arguing for and acquiring such resources.

Whether or not the imperfections of CMM are significant, one cannot deny its huge impact on the software community. This impact can be split into two categories: the indirect or less tangible influence of the capability maturity model, and the process improvement activity that it has generated.

1) Influence of the Capability Maturity Model

   We found, in the organization with which we collaborated, that the SEI work had been a key factor in the shift towards the realisation that process is important, and was one reason why we were able to persuade them of the potential of process modelling.

2) Process Improvement Programmes

   There has been much theoretical work on process modelling and process improvement, but little use within an industrial context. Therefore, the fact that many companies are actually using the SEI approach as part of a process improvement initiative is a success that many process modelling initiatives cannot claim. Furthermore, this success in getting organizations to try using process technology is enhanced by the fact that a number of authors have cited encouraging results using the SEI approach [166-169].

2.3.3 Software Measurement (Metrics)

We do not intend to discuss the merits of software measurement as a discipline in its own right, rather we wish to examine briefly the work aimed at combining process modelling with software measurement ideas. Some of the lessons learned in software measurement work may be applied to process modelling. There appear to be two main reasons for believing that software metrics and process modelling are complementary disciplines:

1) Modelling can provide a framework for measurement, and specifically for data collection.

A process model could include data collection points, thus providing guidance about what to measure, when to measure, and how such information will be used [16-18, 51-53, 170]. In addition, the increased standardization of process models and descriptions (for example, by providing templates for process users) may lead to easier process assessment, reducing the variability that can affect prediction [34]. These co-operative aspects of process modelling and software measurement are echoed in Krasner's suggestion that 'measurement and analysis models' should evolve 'with the process model' [171].

2) Measurement can provide quantitative evidence of the worth of software processes [51-53].

Ambriola believes that measures can actually 'change your view of the process' [13] [173]. Our research endorses this view, and we believe that the combination of measures and models can lead to insights not available by modelling alone.

A significant approach is Basili and Rombach's Goal, Question, Metric (GQM) paradigm [174, 175], which emphasises the need to begin measurement selection by deciding on the goal of the measurement work. A hierarchy links the goal through questions to appropriate metrics or measures. A simplified example of this is given for a previous measurement project [176], which looked at code review effectiveness.

Goal: To improve (active purpose) the effectiveness of the code review process (object) at Site X (environment) from the view point (perspective) of the software engineers.

Q1: How can we measure the effectiveness of the review process?

Q2: How can code reviews be compared?

Refined (Quantified) Goal

To reduce the number of errors in post review code.

Q1: Do we need to know about absolute errors or errors as a ratio?

M1: Possible effectiveness metric

Effectiveness = R / (R+T+C)

where:

Errors found in review	=	R
Errors found in testing	=	T
Errors found by customers	=	C

We believe that the need to start by concentrating on goals can be usefully applied to software process modelling methods[14] (see our example in the following

---

[13] This is a view which was expressed at the experience session of EWSPT'94 (see [172])

chapter). Tate describes a case study which successfully used modelling for a specific purpose (to examine and measure re-work in application development using CASE), and which used a goal-based approach to selecting models and metrics [18]. He states that:

'The emphasis on goals is critical. There is not only an infinite spectrum of software process models; there are also many different software metrics available'.

Another important aspect of Basili and Rombach's work is that they emphasize the need to characterize the environment as a first stage in the measurement programme. Process modelling and metrics can be combined, by using process modelling for this first characterizing step [176]. The ami (application of metrics in industry) method [155] characterizes organizations according to the SEI framework, while (as we noted in examining capability assessment) Pfleeger uses maturity to decide which measures are appropriate for an organization to collect[15] [156, 180, 181].

Thus, combining process modelling and measurement ideas enables us to:

1) Use the goals of our study to help determine appropriate strategies, measures and models.

2) Provide an explicit and visible data collection framework.

3) Enhance the models of software development by providing a quantitative dimension.

2.4 Current Work

2.4.1 Classification Schemes and the Meta-Process

2.4.1.1 Classification Schemes

Numerous process modelling approaches have been proposed during the last ten years or so, and a number of authors have attempted to devise classification schemes for these approaches [74, 182, 183]. Their reasons for doing so include the following themes.

1) To aid identification and evaluation of approaches.

2) To aid comparison of complementary approaches.

3) To aid discovery of parts of the software development process not addressed by existing approaches.

These classification schemes themselves be classified:

---

[14] The realisation that modelling goals are important has led many researchers to suggest that process models should be 'goal' based or goal oriented [177, 178]. However, there appears to be increasing compromise between the goal based and activity based camps with the belief that we need both goals (managers) and activities (users), with some mapping of activities being linked to, or hanging off goals.

[15] Also see [179] which describes similar work.

1) By representation scheme

Many of the proposed classification schemes for process modelling approaches have focused on the representation scheme [182]. However, such schemes can themselves be quite complex. Madhavji needed no fewer than 11 categories to classify the various process modelling notations available [74]. Furthermore, this approach is unlikely to be very robust, in that the inevitable 'new scheme' can easily render the categorisation redundant.

2) By model perspective.

Curtis uses four modelling perspectives (functional, behavioural, organizational and informational [184]) to classify process modelling approaches. His paper also considers the extent to which different representation schemes support these perspectives. Curtis takes the view that specific representation schemes will be appropriate for a given purpose. Thus, by first deciding our modelling purpose and viewpoint, we are much more likely to pick an appropriate representation scheme. By showing how these views are supported by different notations, this classification scheme may be useful not only to academics but also to potential practitioners.

3) High-Level Objectives

Objectives for process modelling approaches fall into four broad categories, corresponding to an increasing desire for process automation: analytical models (mainly for understanding), enaction, simulation, and the desire to build project support environments. One might consider these categories to correspond to an increasing maturity within the subject. That is, we strive to understand development, then produce some enactable models, experiment with possible processes by running simulations, and finally manage to provide higher levels of automated support. However, classifying modelling approaches in this way shows that the attempts to automate the process [107] and to produce environments often pre-date studies aimed at better understanding [10].

What do we gain by categorizing the modelling approaches?

The 'perspective' approach (see Curtis above [184]) maps notations to views, helping us to pick an appropriate notation for a given view. For example, if we know that we wish to focus on information, we can identify notations to support this. However, the categorisation provides less guidance about how we might map to the purpose of the modelling - something which Curtis considers necessary in choosing appropriate notations. This need to consider purpose in order to choose notation is one with which this research firmly agrees, and it is a theme which we will develop later.

A less obvious use of categorization is to support tutorials or introductions to the subject. Such frameworks typically encompass the majority of work to date and provide the beginner with a coherent and organized summary of current work and ideas[16]. Unfortunately, the majority of classification schemes appear to provide little other benefit apart from this educational use.

---

16 The above mentioned paper by Curtis et al [184] is an excellent example of this.

2.4.1.2 Meta-Process

Related to the work on classification schemes is the concept of the meta-process. This term is used to imply that we are looking at a level of abstraction above that of modelling the software development process. As a minimum we wish to have a method for process modelling [185], and the meta-process concept is useful for clarifying this need. In addition, the meta-process idea has been used to provide a framework in which terminologies and classification schemes can be incorporated, and which allows comparison of process modelling approaches, clarification of ideas and so on [88-90, 186, 187]. However, the meta-process concept has been extended to encompass all of process modelling, including developing and evolving the model of the software production process, the support technology and the model of the meta-process itself. To avoid an infinite level of abstraction, we now need notations with reflexive capabilities[17] [188]. The meta-process concept seems useful in reinforcing two other concepts:

1) The need for a support process.

2) The need for evolution of the software process.

Despite this, we wonder whether this additional terminology really serves a useful purpose outside the academic environment. We need methods for process modelling, but it may be more appropriate to develop them fully before we decide how to fit them into categories. For example, the Cookbook approach to modelling methods [32, 189, 190] may provide useful guidance to potential practitioners, and yet it does so without having to consider meta-process. Our own experience is that the term 'meta-process' only serves to confuse, and those with whom we have collaborated see quite readily the need for methods, for support, and for continuing evolution of the process model without need for such terminology.

2.4.1.3 Categories and the Meta-Process

Though there are some benefits to work on categorizing approaches and the meta-process, we wonder whether some of this categorizing is rather premature. There is still precious little experience of applying modelling approaches or modelling methods to industrial settings, and one might, therefore, expect that empirical work would be the priority. Furthermore, without such experience much categorizing may be somewhat superfluous, in that we may be considering theoretical problems which in reality never arise. For example, do we really need to have process models which can be dynamically re-configured, or would such an action actually only lead to greater loss of process control? We do not pretend to have the answers to such questions. Rather we believe that once again there is a need for more experiential work. Indeed, we believe that such work is vital in order to support the more theoretical work (as described above) which has already been undertaken.

2.4.2 Importance of Humans

Despite its often highly technical nature many researchers of software development have suggested that it is the sociological and management aspects, rather than the intellectual rigour, which cause most problems [191]. There is now a body of work that examines the way we develop software, by taking a more behavioural [8], sociological [191], or ethnographic [10] view. This is motivated by the belief that we still do not really understand the way people actually develop software [10], and

---

[17] Otherwise, we would need meta-meta models, then meta-meta-meta models, and so on, ad infinitum.

that we should be examining what people really do. However, until recently the human element had been given little consideration. One possible reason for this is that people tend to concentrate on those things that they know about, so that (not surprisingly) software process models contain the kinds of descriptions that are commonly found within software engineering. Consequently, functional descriptions, programming languages, algebraic notations, and the like abound. Likewise, models have tended to describe parts of the process that are already well-understood, rather than the more subtle human interactions.

In the United States, the work of the recently disbanded software research unit (Software Technology Program) of the Microelectronics and Computer Technology Corporation (MCC) has taken what might be termed a field study approach to attempting to understand the software process [9, 183, 192-194]. This has concentrated on three levels, the individual, team behaviour and organizational behaviour. Much of this work suggests that focus on the way individuals work together has been neglected in the majority of previous process modelling work. This is all the more surprising since it has been suggested that the quality of individual programmers has greatest impact on project success [195]. Thus, understanding what makes a good programmer, or a good project team may be of great commercial value.

Curtis et al. propose a layered behavioural model [8] and suggest that development must be treated in part as a learning communication and negotiation exercise. We note two specific findings of this work:

1) The Super-designer

   One in three projects investigated contained a particularly talented individual who became a focus for the development effort. Often this person would act as the main distributor of knowledge (product and application domain) during the project. This person tended to be distinguished by his or her ability to understand the application domain and what the project/product was all about. For example, he or she might have a 'feel' for telecommunications. Typically good communicators, these people often internalize the progress of projects and take personal responsibility for them[18] [196].

2) The importance of application domain knowledge.

   The 'thin spread' of application domain knowledge is seen as a major problem for software projects [7]. It is suggested that this domain knowledge is crucial[19] and that it is often what distinguishes good engineers from bad, and yet there is often little organizational effort to alleviate this problem[20].

---

[18] This is a finding which we confirm from our own research. For one particularly difficult project that we investigated, which was a new direction for the organization, only one person really seemed to understand what the project was about. Everyone interviewed noted his unique contribution, and talked about the way he had 'driven the project alone'.

[19] See also [6]

[20] This is again a problem encountered during the course of this research. Two possible solutions to this problem are 1) training, and 2) improved communication of knowledge in projects. For one of the projects we investigated, outside training in the application domain had been given to engineers, though this was after project initiation. However, on successful project termination an internal session about this domain was hosted, so that in future engineers on other similar projects

The need to consider the human factor is gaining credence, even with those outside the behavioural camp. For example, the idea of process buy-in or process ownership is one with which most authors agree[21] [197]. Both Tully and Lehman argue that people must feel that they own the process and that productivity gains may be made as a consequence [54]. Indeed, the need to consider the importance of people was one of the strongest themes at the recent European Workshop on Software Process Technology [198]. However, despite this growing interest, the kind of sociological studies described here still seem to fall outside the boundaries of the majority of process modelling work. One reason for this may be that it is very difficult to persuade organizations of their worth. It is perhaps much easier to sell them a modelling method or a modelling tool, than the idea of 'looking at their process'. Nevertheless, this examination of what developers really do is clearly very important, and again suggests a need for more observational, empirical or industrially-based process modelling research. Indeed, simple observational studies may reveal many useful insights. For example, recent empirical work [199-201] has found that many software engineers spent a large portion of their time waiting for important project artifacts which they needed to further their own work.

2.4.3 Multi-Paradigm Approaches

The idea that models should have multiple viewpoints is not a new one, and the approach can be seen in many systems analysis methods [202]. Multi-view models allow us to separate out system complexity by distributing different aspects of the system under study across a number of views. This effectively means that we can have increased modelling capability without models becoming unreadable[22]. This concept has also been transferred to software process modelling [49, 171, 185, 207, 208], one argument being that multi-paradigm approaches represent a way to deal with the problem of conflicting model requirements [184, 209].

A good example of this kind of work is the use of STATEMATE for software process modelling [48, 50, 208-212]. This is a particularly interesting example in that it uses an existing CASE tool [213] rather than a new modelling notation. In this respect it can be considered as a pragmatic approach (see following section). However, this approach of using existing notations is by no means the norm. Despite the fact that most new multi-view approaches actually use the same basic views provided by existing analysis methods (functional or procedural, state or behavioural, data or informational, and occasionally organizational [184]), their proponents still appear to be persuading potential users that the new method or tool is necessary for the process modeller [37, 38]. We echo the views of Harel who, in

---

would have a better initial understanding. Staff at the organization suggested two problems with getting this kind of training. Firstly they had found it difficult to get someone who would pitch the training sessions at an appropriate level. Secondly they considered that the relevance of many points made in the training was only really clear if one had already struggled with such concepts to some extent, i.e. had some project experience. An accidentally-discovered mechanism for communication of domain knowledge was review meetings. One software project manager related his experience of a product proposal review which owing to the lack of understanding of the majority of its participants had become a learning session. He felt that, as a result of this meeting, they all had a much better idea of what was going on.

[21] This was an important factor in the research which we have carried out.

[22] Note that some consider that this shifts rather than solves the problem of complexity, and have suggested the need for distortion oriented presentation techniques [203] (for example fish-eye views [203-205] or perspective walls [206]).

arguing for more flexibility and cross-use of methods, states that 'one of the most unfortunate trends has been in presenting a method as exclusive' [208].

The multi-paradigm approach clearly offers a way to describe the differing aspects and viewpoints of the software process, which no single paradigm can achieve without excessive notational complexity. Though separating out the views of process should simplify models, such methods can still be quite complex and can require significant tool support. In addition, the integration of these multiple views, in order to provide a coherent picture is far from trivial, and again specific tool support may be required (e.g. STATEMATE [212]). Furthermore, for analysis methods there is an implicit assumption that these multiple views are needed in order to complete implementation. For process modelling this may not be the case, and what views are needed should be dependent on the use to which the model is to be put. For example, we may be interested only in a particular view of the process (e.g. procedural) and thus to use other views may involve unnecessary resource.

2.4.4 Pragmatic Approaches

In contrast to complex multi-paradigm approaches, pragmatic approaches tend to adopt a simple usable modelling technique that may not be able to describe all aspects of the process. Tate describes a successful study where, rather than opt for a complex dedicated process modelling approach, an existing CASE tool was chosen for its ease of use and familiarity [18]. Typically, pragmatic approaches use tried and tested notations, such as structured analysis techniques [214, 215], offering an acceptable 'way in' to process modelling for the organization concerned [216]. For example, Starke [217] states that some of the advantages of adopting a structured analysis-based approach are:

1) It is well known and already has agreed usage guidance and terminology.

2) It is a pragmatic approach. Users will find it easier to accept than multi-paradigm approaches or highly mathematical approaches.

3) It allows generic models.

4) It is graphical (hence, readable and usable).

5) It allows for any desired level of granularity.

Starke's concern with the lack of work in applying modelling languages and notations to 'real process modelling problems', appears to be the main motivation for adopting such an approach. He notes that there is 'severely limited process modelling experience' [218] and contends that the need to model real industrial processes is an urgent research issue.

There appear to a number of ways in which approaches could be considered pragmatic:

1) There are approaches that opt for a single paradigm [36].

2) There are approaches that (though they may have multiple views) use tried and tested (readable and usable) notations [208, 217]

3) There are approaches that opt for both a single paradigm and a known usable notation [215].

Clearly there is some overlap in what can be considered a pragmatic approach. We simply cannot say, for example, that a muliti-paradigm approach is not pragmatic, or that a single view approach necessarily is. Pragmatism is more about the attitude taken by the modeller and the accessibility of the technique. Though they may differ in the exact notation or modelling approach taken, the common theme in all of these approaches is that they prefer to sacrifice some of the modelling capability in favour of simplicity and usability[23], in an attempt to be more acceptable to potential practitioners. Thus, pragmatic approaches appear to offer a way to address the 'urgent need' for industrial experience.

2.5 Problems, Issues and Concerns

2.5.1 Current Issues in Process Modelling

Here is a brief list of some of the issues which are still being debated by the process modelling community. We have added to these descriptions of the issues some discussion of the direction which we believe the modelling community may take.

1) Enaction: Do we need enactable models?

   Enaction can help with getting the right information to users, and aiding complex sequences of actions (as with the Unix 'make' facility). A number of tools provide these enaction capabilities. However, as we have noted, this capability is often neglected. Enaction is felt increasingly to be useful in supporting co-ordination [140]. Certainly those support tools using enaction often employ it to co-ordinate large sequences of well-understood activities (e.g. Process Weaver [41]). Indeed, this seems to agree with some of the lessons learned from the office automation community [10], notably that repetitive tasks can be supported by tools but that attempting to automate more creative human activities and interactions is less successful.

   There is some confusion about terminology, in that enaction has become almost synonymous with automation, particularly in Europe. Thus, the apparent debate about enaction may really be one about where we need formality, and where we need automation.

2) Formality: Where is it needed?

   There is increasing agreement that not all activities can be formally defined. We appear no longer to be arguing about whether we need formal and executable models (as proposed by process programming), but about where formality is or is not appropriate [93].

3) Understanding versus Executability

   Is it more important to have understandable notations or executable notations?

   In our view such questions cannot be answered out of context; the purpose of the modelling, the environment, and the users all have a bearing on what

---

[23] Some have voiced even stronger anti-notational views, for example Pyzdek [219] suggests that the key to process improvement is 'keeping it simple', and rather than concentrating on what notations to adopt he suggests that what 'really counts' is genuine commitment to, and involvement in the process improvement program.

the notational capabilities should be. However, any notation chosen will be worthless if the users cannot understand and use it since 'people are important' [198]. Hence, while attempting to take into account both factors, we believe that a heavier weighting should usually be given to usability when using process modelling technology.

2.5.2 Critique of the 'state of the art'

We now summarize some of the positive and negative aspects of the process modelling work which we have discussed in this chapter.

2.5.2.1 Positive Points / Progress

1) The work on process modelling has led to a better understanding of some aspects of the software development process. The recognition of iteration as being an unavoidable reality is a significant example of this.

2) Though there is no empirical evidence that process does have an effect on quality, most organizations seem to accept that it does. The enthusiasm with which process maturity has been adopted, is testament to this. Though this is worrying from a scientific point of view, it does mean that process modelling is a technology that organizations may be willing to try.

3) The importance of the role of humans in the software process is beginning to be recognized. For example, there is increasing realization that we need to understand the way people work, rather than simply impose processes upon them, and process ownership is now seen as an important consideration.

4) There are now a mass of possible notations and descriptive methods for the modeller to use. These include graphical notations offering visibility and ease of understanding, and detailed notations which can in some cases be executable.

5) There are already frameworks, classification schemes and agreed terminology in place to enable researchers to compare process modelling approaches theoretically.

6) There is increasing agreement that we cannot use a model to control the process, and that the role of process modelling is to provide understanding, guidance and process support.

7) Multi-paradigm approaches offer the capability to be able to model many aspects of the development process, and to combine these views into a coherent framework.

8) There exist impressive process modelling tools, which offer graphical interfaces and modelling languages, as well as process engines to execute notations.

9) A number of researchers have successfully used process modelling (or process maturity) in conjunction with other software engineering initiatives (notably software metrics and CASE).

2.5.2) Negative Points / Concerns

1) There is still relatively little understanding of the way people develop software, and yet there is little work on observing real software processes.

2) There is some empirical evidence that process modelling is a useful industrial technique. However, we do not yet understand which techniques are best or what aspects of models are most helpful for a given context.

3) There has been little attempt to focus on specific project or organizational goals.

4) There has been little industrial use of the techniques, and even less reported experience.

5) There is little to guide the potential modeller. There are no 'methods' that aid selection of appropriate process modelling notations.

6) There appears to be far too much focus on automating the development process. It is significant that the capabilities of the tools produced are often under-utilized in practice. Users are reluctant to allow instantiated process models to control their work.

7) Despite the fact that some have suggested a more pragmatic approach to modelling, there are many organizations for whom the scale of many existing techniques is inappropriate.

2.5.3 Need for further work

We once more draw attention to the high number of unresolved issues in software process modelling, which is indicative of a subject in its infancy. Despite all of the work on software process modelling we still don't really understand the development process, and there is very little reported evidence of empirical industrially based work.

We suggest that what many organizations actually need is a model which though it may be imperfect, is usable and useful. Furthermore, we suggest that sufficient notations already exist, and that the real problem is that not enough work is being done in applying process modelling to real world problems. This industrially-based modelling work is particularly important if we are to gain much needed experience in observing and modelling 'real world' processes. Indeed, much of the theoretical work which has been discussed in this chapter, can only be validated or justified by future empirical study.

Finally we note the comments of Tamai who re-iterates Rodden's [10] call for observing what people really do:

'Whatever approach may be taken, the observation and understanding of real software processes should be the basis for constructing an appropriate process model'. [220]

2.6 Work of a similar nature to that proposed by the research

Those few who try process modelling in an industrial environment appear to be taking a simple or pragmatic approach to modelling, using existing systems analysis notations. For example, Tate [18] and Pengelly [215] adopt a data flow based approach to process modelling[24] and McGowan and Bohner [197] use IDEF diagrams. In distinguishing our work from these authors it is important to note that this is a small grain distinction, and that our approach and rationale are very similar. Indeed, there is an astonishing paucity in this area, and still very little empirical work has been published[25]. The majority of the process modelling community seems to be more interested in devising ever more complex tools and classification schemes, than in applying the technology to real problems. Thus, we have much in sympathy with others who are attempting to put process modelling into practice.

Our exploratory work used an existing notation (data flow diagrams) for process modelling. This initial study, though not significantly different in the modelling notation or strategy from the work cited above, does introduce guidelines for choosing appropriate modelling notation. (We will consider differences in notation, and why we chose our particular notations in our discussion of modelling methods in chapter three). We have used process modelling to focus on a specific problem (similar to the work of Tate [18]) and the lessons learned from our work are similar to those of McGowan and Bohner who also contrasted the theoretical and actual development processes.

However, having used our initial study to demonstrate the utility of process modelling to the collaborating organization, we then extended our work. Our later work is distinct in a number of ways:

1) We have used our process model as a data collection framework.

   We used the process modelling to discover the invariants in the process, and then produced a mechanism to collect data based on these invariants.

2) We have combined the process model and the data collected such that the model serves as a mechanism for displaying that data in an accessible way.

3) We have used our (combined) process model data to link process activities with their impact on projects.

4) We have extended the data flow approach to develop a unique notation which allows us to depict the process of five projects over time.

   For example, rather than allow for iteration, sequencing etc., the notation shows the true extent of iteration, by showing not how many times an activity is revisited, but when, how much effort it took, and what was the average distribution of effort during that time.

---

[24] Interestingly both of these authors are also keen to emphasize the link between process modelling and software measurement.

[25] This was a strong concern at the recent European Workshop on Software Process Technology, where it was felt that owing to this research deficit any reported experience was made much more valid and valuable to the community as a whole [172].

5) Our modelling approach has allowed us to uncover unique findings about the nature and extent of project variation at a TickIT accredited [59] software developer.

The following chapter further describes our research approach, both in terms of the research methods and modelling methods adopted.

3. Research Approach

Chapter Synopsis

This chapter considers the two main elements of our research approach. In section one we examine the choice of research methods available, before noting how our own work can best be described as case study research. In section two we examine the choice of modelling notations available. Owing to the multitude of possible notations we concentrate on large-grain differences. Therefore, we examine in detail three very different kinds of approaches which we consider to be representative of the state of the art. Finally we examine the reasons for the choice of notations which we adopted for our own work.

3.1 Research Methods

Hammersly [221] notes that one of the problems in describing research methods has been the 'widespread tendency to see research method in terms of contrasting approaches or paradigms involving different epistemological assumptions'. Thus, for example, in the 1920s and 1930s the case study was often contrasted with statistical work, in the way that qualitative and quantitative methods are still compared. Hammersly argues that much of this categorization is artificial, for example, that case studies and surveys are not really separate approaches, more a shift in emphasis, to make the method more appropriate to given conditions. Nevertheless, the widespread use of such terms as experiment, quasi experiment, survey, case study and ethnography provides a useful framework for consideration of research approaches. However, it appears that there is still some disagreement as to exactly where the boundaries between methods lie. Thus, Smith [222] considers case study analysis methods as a subset of quasi-experimental design, whereas Yin [223] makes a clearer distinction and sees the case study as completely separate. The following sections attempt to describe the core aspects of the various research methods, and to note their applicability or appropriateness for our own study. We start by describing experiments and quasi-experiments, including a brief discussion of the concepts of reliability and validity, to which we will occasionally return in discussing other approaches. We then describe surveys, case studies and ethnography. Despite the fact that we identify where our own work differs form the case study approach, we believe that this particular method still offers the greatest potential as an appropriate framework for our kind of on-site study. Thus, our discussion of the case study is the most lengthy of our 'methods' sections. We initially describe one view of case study research at length, and then go on to give some conflicting descriptions of the case study. Finally we discuss the range of work which has been banded under the heading of ethnography, concentrating specifically on the recent attempts to use this approach to investigate the software development process [191].

3.1.1 Experiments and Quasi-Experiments

In the pursuit of software engineering as a scientific discipline one method of evaluation of new technologies and approaches would be by experiment. We will examine the features of research methods that typify experiments and assess the

appropriateness and viability of these features for our own process modelling research.

Hammersly [221] believes that the chief distinction between experiments and other research strategies is that 'the researcher creates the cases to be studied throughout the manipulation of the research situation thereby controlling theoretical and at least some relevant extraneous variables'. For example, a classical scientific experiment is designed such that independent variables are varied systematically, and objective dependent variables are then measured.

Adelman [224] lists five 'basic components of most factorial experiments'. These are:

1) Participants

2) Experimental conditions or independent variables.

3) The tasks the participants perform.

4) Dependent variables.

   Adelman states that 'Objective measures (e.g., decision quality), observational measures (e.g. decision process quality) or subjective measures (e.g., user confidence) can all be used as dependent variables.'

5) Procedures governing the implementation of the experiment.

   For example, in order to be able to say that an improvement is due to some factor we need to be able to control for 'plausible rival hypotheses'.

The two chief concerns of the experimental design (and indeed of most research design) are to preserve reliability (replication) and validity. We will discuss these issues here, though it should be noted that these are design issues which are relevant to any research method, and to which we will return later.

3.1.1.1 Reliability

For an experiment to be reliable it should be able to be repeated such that it would give the same results. The chief mechanism in ensuring experimental reliability is to modify the experimental procedures until the same result is gained when the procedures are applied to the same situation. In other words, the procedures are modified until the experiment and the result are repeatable.

Clearly where each run of the experiment constitutes a study of some length, we cannot always afford the luxury of modifying procedures in this way before commencing the experiment 'for real'. This suggests that the experimental framework may not be appropriate to process modelling work such as ours. However, one approach to this problem has been to attempt to 'prototype a process experiment' [199]. This approach sets up an experiment and collects data from it, but recognises that the experimental design can be changed as part of a longer on-going experiment.

A further problem with the experimental framework in social situations is that we do not ever have the 'same situation', and thus it is difficult to justify claims about the ability to replicate results elsewhere. Indeed, for process modelling in industrial situations, many would claim that the conditions at their site are 'not like anywhere else', and this is a significant factor in suggesting that the experimental framework

is inappropriate. However, we can still use formal experiments to test the effects of methods or tools within an organization. In this case we can still use an experiment if we are able to control over our variables. Pfleeger [225] gives an example of testing a hypothesis that a new programming language will effect the quality of code. In this case we would design an experiment so that we looked at a number of projects which used different languages, but where other variables, e.g. project team experience, project difficulty, application domain, were the same.

3.1.1.2. Validity

Internal Validity is concerned with 'establishing a causal relationship', that is establishing that it is the independent variables and not some other factors (or variables) that have caused the effects we have measured to our dependent variables. These other factors are often considered as 'rival hypotheses', and their are two main ways in which experimenters attempt to control for these 'rival hypotheses'. The first solution is to design experiments such that the effect of the rival hypothesis can be judged. There are two main objections to this approach. The first is really a practical objection, in that we often have a large number of independent variables, and thus inadequate resources to check all for rival hypotheses. The second objection, is that we can never really know all plausible rival hypotheses; there may always be alternative explanations (hence, we always attempt to refute null hypotheses rather than confirm hypotheses). The second solution is to attempt to eliminate the possibility of rival hypotheses being able to affect the results, typically by randomizing. Thus, we might wish to choose subjects at random, projects at random and so on. However, this may not always be possible. For example, in our own work we wished to choose both successful and unsuccessful projects. This means that a quasi-experimental approach would be needed, such that each threat to internal validity should be noted and ruled out in turn.

Construct Validity is concerned with ensuring that the effect of a variable is not confounded with other experimental conditions. In other words we want to be sure that the effect we are seeing is due to the variable and not some other aspect of the experiment. An example is that many people will feel better in a clinical trial, even if given a placebo. Thus, the effect is a combination of the psychological effect of the treatment and the physiological effect of the medication. In the medical example, we have to compare the results of the trial against those of the placebo. For a process modelling study it is quite possible that there are a number of threats to construct validity. However, we can only eliminate these threats if we have some idea of what they might be. In this case we can either attempt to minmize the influence of the threats or to design the experiment such that we can assess their effects.

3.1.1.3 Quasi-Experiments

There are many forms of study which can be considered as quasi-experimental. Broadly these designs occur when an experimenter cannot control experimental conditions as fully as for an experiment but still wishes to use the framework of an experimental design for the data collection. For example, simple interrupted time series designs use the group being tested by repeatedly testing their performance before and after the introduction of the factor under scrutiny. Adelman [224] examines three types of quasi-experimental design, namely:

1) Time series designs,
2) Multiple time series designs using a control group and
3) Non-equivalent control group designs.

In order to point out their advantages and disadvantages he [224] also discusses three versions of the 'inferior' 'pre-experimental designs', as a comparison.

Pre-Experimental Designs

1) The One-Shot Case Study (or the One Group Post-test Only Study).

This is where 'one unit is given the treatment...' [224]. There is no pre-testing and there is no control group. Inferences are based upon 'general expectations of what the performance (data) would have been' [224]. This method threatens all four types of validity; (i) it has no control over internal validity, (ii) it has no measurement and, therefore, it is impossible to gauge the effect of extraneous factors, (iii) it has no measurement of performance variables, or comparison with another group; consequently it is impossible to asses statistical conclusion validity', and (iv) there is no basis for predicting the effect of the 'treatment' on another group.

2) The One-Group Pre-test/Post-test design.

As above but with pre-testing. The main problem with this design is that it 'does not control for the effect of other plausible hypotheses..' [224].

3) The Post-test Only Design with Non-equivalent groups.

Compares the post-test performance of the group under study (e.g. having received some treatment) to another different group (e.g. who did not receive the treatment). Unfortunately because the groups are non-equivalent and there is no pre-testing there is no way of showing that they would not have differed in the same way even if the treatment had not been applied.

Quasi-Experiments

1) Time-Series Designs.

Considered appropriate for 'use by as few as one group' [224]. 'The 'simple interrupted time series design' uses the group itself as a partial control for alternative hypotheses by measuring the group's performance repeatedly both before and after treatment intervention. It is considered a weak design 'because of a number of threats to its internal validity' [224]. Examples of such threats are:

- that some event other than the treatment caused the change,
- changes in instrumentation (record keeping or administration),
- selection and changes in the composition of the group.

2) Multiple time series designs using a control group.

Adelman notes that the addition of a single control group does not remove the threat to external validity.

3) Non-Equivalent Control Group Design

The Non-Equivalent Control Group Design 'adds a pre-test measure to both groups in an effort to control for factors, other than the treatment..' [224], but uses 'only one pre-test and one post-test observation per group' [224], hence, it is 'not as effective for controlling internal validity threats' [224]. The design controls for all but four threats to internal validity, namely:

(i) Selection/maturation bias: Members may change from one group to another.
(ii) Instrumentation, e.g. scaling problems.
(iii) Statistical regression to the mean.
(iv) Local history.
All of the above threats are a 'function of selection bias' [224], and a result of a lack of design randomization.

3.1.1.4 Appropriateness of Experiments for our work

Experiments are chiefly concerned with assessing the impact of some specific variable or variables. Their main advantage is that they offer greater control for the researcher. However, this is paid for both by a lack of flexibility and by the fact that the experiment may render the situation artificial. For example, in examining social situations the level of control required for an experiment may not be representative of the situation being observed. Therefore, experiments are ineffective where we wish to gather information about naturally occurring phenomena or situations over which we have little control.

Though we have noted some concerns with experimental design for process modelling our main reason for rejecting this approach is that we are not trying to carry out this kind of quantitative assessment on process modelling. Our work is chiefly concerned with observing the realities of process modelling, and attempting to show that a simple approach can be utilized by a typical software engineering organization. We are not attempting to demonstrate process improvement, nor to show that the introduction of the new technology has led to some marked increase in productivity. Thus, we must look to other research methods for an appropriate strategy.

3.1.2 Surveys

Surveys involve selecting a relatively large number of naturally occurring cases. Surveys have traditionally been associated with more 'hands off' studies (e.g. postal surveys) than case studies which have usually involved far more work in the field. For example, survey methods are often used to ascertain the views of some representative sample of the general public. Whilst surveys typically involve more detail about each case than experiments, they have also have less information on each case than a case study. Hammersly [221] argues that the difference between case studies and surveys is one of degree: 'We have a gradient or dimension here not a dichotomy'.

Thus, choosing a survey selection strategy involves making a trade-offs between the detail (and likely degree of accuracy) in each case against the number of cases.

3.1.3 Case studies

As we shall see from the following section there is no general consensus as to the exact meaning or definition of a case study. Thus, in order to give a flavour of the characteristics of case study we shall first draw mainly from one source, namely Yin's description of case study research. [223]. Yin defines a case study as:

'...an empirical inquiry that:
- investigates a contemporary phenomenon within its real life context, when
- the boundaries between phenomena and context are not clearly evident ; and in which
- multiple sources of evidence are used.'

However, though this definition would seem to fit perfectly with the investigation of process modelling on-site, the meaning of the term has been further refined, not least by Yin himself so that in order to qualify as a case study such work must also be designed, organized and carried out in a more controlled manner (see section 3.1.3.1 below). We now briefly examine some of the advantages claimed for the case study and summarize some of the important elements which contribute to the case study method.

3.1.3.1 Characteristics of the Case Study Method

The major analytical difference between the case study and experimental framework is that whilst experiments sample over their state variables (e.g. an experiment would examine a variety of projects) the case study samples from the state variables (e.g. a case study would usually examine typical projects, though cases may also be chosen precisely because they are atypical). Thus, the case study will typically examine one or more cases in some depth. The case study is, therefore, often useful in social settings and Yin suggests that it should be 'preferred in examining contemporary events, but when the relevant behaviours cannot be manipulated'. This fits well with the situation of examining software development where the research is often far less important than the primary development goal of producing software, and where the researcher often has very little control over events.

Another strength of the case study is its flexibility. For example,

1) The case study can deal with a number of different sources of evidence,

2) The case study can be used within or to subsume other studies (e.g. a survey within a case study, or a case study which contains experiments).

3) The case study can also include 'and even be limited to qualitative evidence' [223].

3.1.3.1.1 Case Study Design

Case studies can be characterized along two orthogonal axes, single or multiple case, and holistic or embedded designs. However, irrespective of this the design of a case study design should always consider five categories or components. These are:

1) The questions which the study hopes to address.

2) The propositions which the study hopes to investigate.

   Yin notes that 'some studies may have a legitimate reason for not having any propositions'. He calls these kind of studies explorations. He then states:

   'Even explorations, however, should still have some purpose. Instead of propositions, the design for an exploratory study should state the purpose as well as the criteria by which an exploration will be judged successful'

3) The units of analysis of the study - what the 'case' is.

4) The logic which links the data back to the propositions.

5) The criteria for interpreting findings.

However, note that Yin says of case study findings 'there is no precise way of setting the criteria for interpreting these type of findings.

According to Yin [223] all case studies should also have a defined protocol to increase the reliability of the design. He suggests that this protocol should include the following four sections:

1) Overview

   Includes the rationale for the selection of the site for the study, any propositions or hypotheses to be investigated, any theoretical or policy references and the purpose and setting of the study.

2) Field Procedures

   Guidelines for coping in the field (most relevant where there are a number of case study researchers, who may require some training). Field procedures emphasize strategies for gaining access to the organization, scheduling collection of data, coping with unanticipated change (e.g. lack of availability of staff) and the like[26].

3) Case Study Questions

   These questions are intended to 'keep the research on track', and often are accompanied by a list of possible sources of information. Note that these are questions for the interviewees, the case or multiple cases, but not questions for the entire study. For example, though the case or cases may be designed to answer some specific question or questions (case study questions) the whole study may wish to answer some more general question.

4) Guide for the Case Study Report

   Note that again this is only intended as a guide. Yin states that:

   'In fact case study plans can change as a result of the initial data collection, and investigators are encouraged to consider these flexibilities - if used properly - and without bias - to be an advantage of the case study strategy'

3.1.3.1.2 Conducting the Case Study

In conducting the case study itself there are said to be three 'overriding principles' of data collection [223]. These are:

1) Having multiple sources of evidence.

   Sources of evidence within a case study may be of a number of different types, including documents, archival records, interviews, direct observation, participant observation and physical artifacts. The key to the multiple sources idea is that evidence should come from two or more sources but converge on the same set of facts or findings. For example, our investigation of the software process will utilize procedures documents, documents produced by the process (physical artifacts), and interviews in

---

[26] NB For our research all of these things happened and were all decided by the single researcher working on his own initiative. However, this is because this person had control over the direction of the research, and this is not usual for this kind of study.

order to investigate whether there is a problem with the perception of the process. Similarly we will use e-mail (records), time-sheets and focused interview in order to collect information about the effort spent on various project activities.

2) Having a case study database.

The idea here is to have a collection of evidence that is separate from the case study report. Typically this evidence consists of notes (on interviews or analysis of documents) survey or other quantitative data and some bibliography of documents (in our case it would also include the process models produced at various stages of the study). This collection should be retrievable so that in principle other investigators could examine the same evidence, and assess the findings. Therefore, the classification scheme adopted is felt to be unimportant, as long as it is accessible to an outside investigator. However, the evidence need not be re-written in order to make it more presentable, and indeed if the notes and the like are readable it may be preferable to leave them intact, as they were produced at the time.

Another interesting form of evidence is 'narrative'. Here the investigator has a much more open ended protocol and uses the narrative to document the connection between pieces of evidence and issues in the case study.

It is worth noting that 'with case studies the distinction between a separate database and the case study report has not become an institutionalized practice' [223], and that there is no consensus about what form the database of evidence should take. Furthermore, a problem with industrially-based studies is that the evidence may be considered confidential. For example, in our case the organization may not wish their competitors to know details of their process, or the views and criticisms that its own employees have about that process. This makes the presentation of evidence much more difficult in that the investigator is forced to just 'report' findings rather than being able to 'point' to the evidence.

3) Having a chain of evidence.

The investigator aims to have explicit links between the questions asked, the evidence and the conclusions.

3.1.3.1.3 Analysis of Case Study Evidence

Case studies usually adopt one of three common major analysis techniques[27]: pattern matching, explanation building or time series analysis, or one of three 'lesser techniques' analysis of embedded units, repeated observations or case surveys. Though we do not intend to describe these techniques here, it is important to note that these common techniques do not attempt to show that the findings that they generate are statistically generalizable. Rather, in keeping with the case approach they suggest theories, and it is these theories which may be generalizable.

However, examination of the use of these analysis techniques can be useful in pointing out what is and what is not a case study. For example, Yin [223] in describing embedded analysis notes that:

---

[27] We note the more recent development of specific case study analysis techniques for software engineering in the following section.

'If the embedded unit is itself the main focus of attention (or is allowed to become so) and if the larger case is only a mere contextual matter, the effort should not be considered a case study, and some other research strategy should be used'.

For our own work this kind of advice is significant in that it suggests that case study analysis techniques of our study of individual software projects (instances of process) will be inappropriate if these projects are the focus of the case.

3.1.3.2 Pilot Case Studies

In order to refine their strategy the researchers may choose to do a pilot case study. The choice of the pilot study site may be for any number of (often pragmatic) reasons. For example, for this research access to an organization and data was difficult, and approaching a site with which the author had some previous collaboration increased the chance of the pilot (exploratory) study being able to take place. The pilot study allows the researcher to refine the data collection strategy, both in terms of what is to be collected, and what procedures will be used in order to collect it. The pilot study is important in refining the objectives and strategy of any subsequent work. Indeed, the pilot may be carried out 'prior to the final articulation of the study's theoretical propositions' [223].

The pilot report is often mainly for the use of the investigators (or the collaborating organization) and may be documented in a much less formal way (e.g. it may be in the form of memoranda) than the main case study report.

3.1.3.3 Should Case Studies be Representative?

As with the argument over what constitutes the case study, there is also much contention over whether case studies should or need to be representative. This argument itself falls into two camps, those who believe that this irrelevance is temporary, and those who believe that irrelevance is absolute. The idea that representativeness is only temporarily irrelevant has led to the partitioning which regards case studies as either (i) appropriate to exploratory work only or (ii) being made representative by the application of quantitative procedures. However, both of these views hold with the fundamental requirement for representativeness.

A more radical viewpoint is that representativeness is 'absolutely irrelevant' [222]. One reason for adopting this argument is that the case study may be used for a completely different kind of purpose to that which motivates a survey or experimental study. For example, the case study may be a vehicle for description of a phenomenon. A second (more theoretical) reason for regarding representativeness as irrelevant is the rejection of representativeness as a basis for validity. Worsley states [226]:

'The general validity of the analysis does not depend on whether the case being analyzed is representative of other cases of this kind, but rather on the plausibility of the logic of the analysis'.

Mitchell offers strong support for the view that representativeness is absolutely irrelevant in case studies. Mitchell [227] describes a case study as:

'examination of an event (or a series of related events) which the analyst believes exhibits (or exhibit) the operation of some identified general theoretical principle'.

The argument for irrelevance is based on the belief that it is logical inference and not statistical inference that leads us to be able to have analytic generalizability. For example, Mitchell [227] notes that *'logical inference is independent of statistical*

inference' and that *'statistical analysis merely permits the inference that characteristics within the sample may be expected within the population'.*

Mitchell [227] further notes that there is a tendency, particularly with quantitative studies to assume that the logical connection which has been postulated may be assumed to exist in the population if it can be inferred (statistically) that it exists in that population. In other words, statistical inference is being confused with logical inference which is being assumed. The fact that logical and statistical inference are separate brings us back to the argument about the irrelevance of representativeness. The inference from case studies is logical or causal and not statistical (i.e. we cannot extrapolate to a population), relying not on enumerative induction but on analytic induction, and thus, whatever argument we use about the representativeness of the case does not have any bearing. Furthermore, the selection of a case should not, therefore, depend on how 'typical' that case is but rather upon the potential for explanation that the case provides.

An example of how it is the plausibility of the logical argument, rather than the statistical inference which allows us to accept causality is provided by the following example. It is reported that a study has shown a statistically significant reduction in asthma suffering amongst condom users. At first sight it would appear that we have only an association (perhaps some anomaly). We would be unlikely to accept that there was a causal link based on this evidence. This illustrates the weakness of inference alone. However, we now bring logic to bear and assert our reasoning. Dust mites feed on (among other things) dried semen, and thus condom use, by reducing a food source for the mites, leads to reduced asthma suffering for the user. It is the plausibility of this explanation which would then allow us to generalize our argument. That is we are able to extrapolate to a larger population because we believe in the explanation.

3.1.3.4 A More Quantitative View

A significant development in attempting to further define the role of case study in software engineering is the DESMET [228] research and development project for evaluating software engineering methods and tools. This project aims to produce guidelines for methods to 'assess whether a method / tool appears to be better than another method / tool'. The case study is thus specified as:

'... a way of evaluating methods and tools as part of the normal software development activities undertaken by an organization.'

Here we can see that the core of the case study ideal of evaluation within the social setting remains, but that the definitions have moved towards evaluation. The more exploratory or investigative case study is not the main focus of the DESMET case study specification, and the method appears more aimed at quantitative assessment than qualitative. Thus, a typical use of the framework would be to provide quantitative evidence of process improvement as a result of the use a particular method or tool.

There are two reasons why our own work does not completely fit this definition.

1) Our own work was more investigative than the above definitions allow. We were not concerned with quantitatively demonstrating process improvement. We were more concerned with investigating the kind of insights (about part of the development process) that process modelling could provide. However, we were also concerned that we should be providing an example of the utility of simple process modelling.

2) The DESMET Case Study Design and Analysis (CSDA) module makes the following assumptions:

'the organization has at least well-defined standards for software development and that adherence to those standards is monitored. In addition, it is assumed that the organization is monitoring and planning individual projects in quantitative terms'.

However, part of the reason for our study was to ascertain just how much adherence to procedures there was within the organization. Indeed, our work found that there was considerably more deviation among software projects than we had initially suspected. Thus, we were unable to make the above assumption.

Nevertheless the DESMET framework does provide clear guidelines for case study design and analysis which are useful even if attempting more exploratory work. We will both use and make reference to parts of this framework within our study.

3.1.3.5 Case Studies: Summary

Despite the clear disagreement over what exactly is a case study, some similarities emerge. Case studies are clearly suited to the examination of phenomena which either occur naturally or which need to be examined with a minimum of disruption or interference. Case studies typically involve the detailed description of a few, or sometimes a single case. They allow us to use a number of sources of evidence, and to use other research methods within the case study. In addition, the case study framework, for example as seen is DESMET, provides sound guidance as to the kind of issues which need to be considered in undertaking on-site or industrially based work like ours.

However, we also note some of the dangers of a case study approach. Smith [222], believes that these stem not from theoretical weakness but from the closeness of the researcher to the phenomena. He notes the difficulty of retaining objectivity, and the difficulty in convincing others of the acceptability of case study research.

This difficulty in convincing others of the worth of case study work is further complicated by the confusion over the need for representativeness. For example, Mitchell [227] suggests that we should select cases based on the potential for explanation that the case provides rather than whether the cases are typical. Worsley [226] also cautions against considering representativeness in the analysis of case studies noting that it is the plausibility of the logic of the analysis which determines its validity. That is, we must be careful to judge our hypotheses on their plausibility, rather than relying on statistical inference, and furthermore, that it is the plausibility of the logical induction (and argument) rather than the representativeness of the case which allows us to extrapolate our arguments to the larger population.

However, in practice we still wish our choice of projects to be representative, since the plausibility of our argument for extrapolating results may itself be weakened if this is not the case. For our own study we wish to select projects which are typical of software development (at least within the site) in order to argue whether our results are generalizable. For example, if we only examined projects of one particular type (say small projects with the same project manager) then our (analytic induction) that all projects will exhibit similar behaviour is clearly a less powerful argument than if we have chosen a representative sample of typical projects. Hence, we will choose cases (in our case projects) which provide sufficient material for explanation, without sacrificing the representativeness of those cases.

Despite the problems with retaining objectivity and with convincing others of the worth of case study work the case study method provides a flexible and powerful framework for the observation and description of those situations (such as software development) which need to be studied with the minimum disruption. Hence, the case study method appears to provide an appropriate vehicle for the design and depiction of our research.

3.1.4 Field Study / Ethnography

Ethnography is usually referred to by social scientists as 'field studies'. The term simply means any studies which take place at an actual work site. Ethnography typically relies heavily on direct observational work, and does not, for example, use questionnaires or interviews, as the feeling is that the questions reveal too much of the prejudices and preconceptions of the interviewer or question setter. In contrast observation reveals implicit social interactions (e.g. status governed interactions) which are not revealed by questions, and which the subjects being asked the questions may not always be aware of.

Central to ethnography is the rejection of the notion of having hypotheses or propositions for the purpose of the study. Indeed, 'one of the key elements of ethnography is that there are no explicit terms of reference as these are seen as inherently prejudicial to a study' [229]. Proponents of ethnography say that you should simply 'get into a culture and observe its practices without any objectives apart from understanding the detail of the culture' [229].

However, within software engineering field studies this idea has been tailored by Sommerville and others, in order to produce more focused studies which reflect the need to understand the process. This focused ethnography has been used in both in studying the software development process [191] and in studies of Computer Supported Co-operative Working (CSCW) [10]. This has produced an approach which is somewhere between standard ethnography and field study, in that it uses some purpose to structure the investigation, but does not have a formal experimental design or use case study analysis techniques.

This 'focused ethnography' approach appears to be particularly appropriate to observational studies. However, it is less useful where we do have some general proposition which we wish to investigate. In our study we wish to do more than observe some facets of software development. For example, we will examine whether simple process models can highlight process problems. Therefore, our work benefits from the additional structure offered by case study methods (see our use of case study design methods in Chapter Four and Chapter Five).

3.1.5 Industry as Laboratory

Potts [230] in examining the failure of research to influence industrial practice suggests that the problem is the prevalence of the phased 'research then transfer' approach and he suggests a complementary approach called 'industry as laboratory'. He argues that the 'industry as laboratory' approach leads to three major changes:

1) 'Greater reliance on empirical definition of problems'. Potts considers that: 'Empirical observation of projects becomes a legitimate focus of research in its own right'.

2) 'Emphasis on real cases'.

3) 'Greater emphasis on contextual issues'.

This approach is said to have the following benefits:

1) 'The definition of the problem to be solved comes more directly from a detailed understanding of the application environment'.

2) 'There is less emphasis on a separate technology transfer stage'.

3) 'As research progresses it becomes increasingly problem focused'.

We have much sympathy with the view that there is a need for more industrially-based research. Indeed, our own work will have many similarities with Potts' 'industry as laboratory' approach. We will take into account the environment and organizational needs in selecting an appropriate modelling notation. The problem (area of process to be studied) came from the organization, the work is to be carried out on-site in close co-operation with users, and the research will be progressed taking into account their needs.

3.1.6 Approach Taken by this Research

Our work will take place in a social situation, where we have little control over the variables, and where we wish to be of minimum impact, but where there are a number of sources of evidence. We believe that experiments and quasi-experiments will not be appropriate approach since we have insufficient control over variables, and we do not wish to impose artificial constraints.

Our work will examine a small number of cases in some depth, by selecting typical projects (sampling from the state variable), and thus survey methods are inappropriate.

In structuring our study we need to use some research design framework in order to be able to examine the utility of process modelling. We do not believe that ethnography provides us with sufficient methods to be able to investigate the use of process modelling.

All of the above point to case study research as the most appropriate method for designing and describing our work. Furthermore, the use of the case study method allows us to incorporate a number of sources of evidence into our work. The following chapters thus describe our work in terms of two case study designs, an exploratory case (Chapter Four), and later instance cases (Chapter Five). In order to do this we have used the DESMET case study design and analysis module extensively, both to help us to understand the issues, and the relevant questions, and to structure our presentation for the reader.

3.2 Modelling Methods

3.2.1 Comparison of Process Modelling Techniques

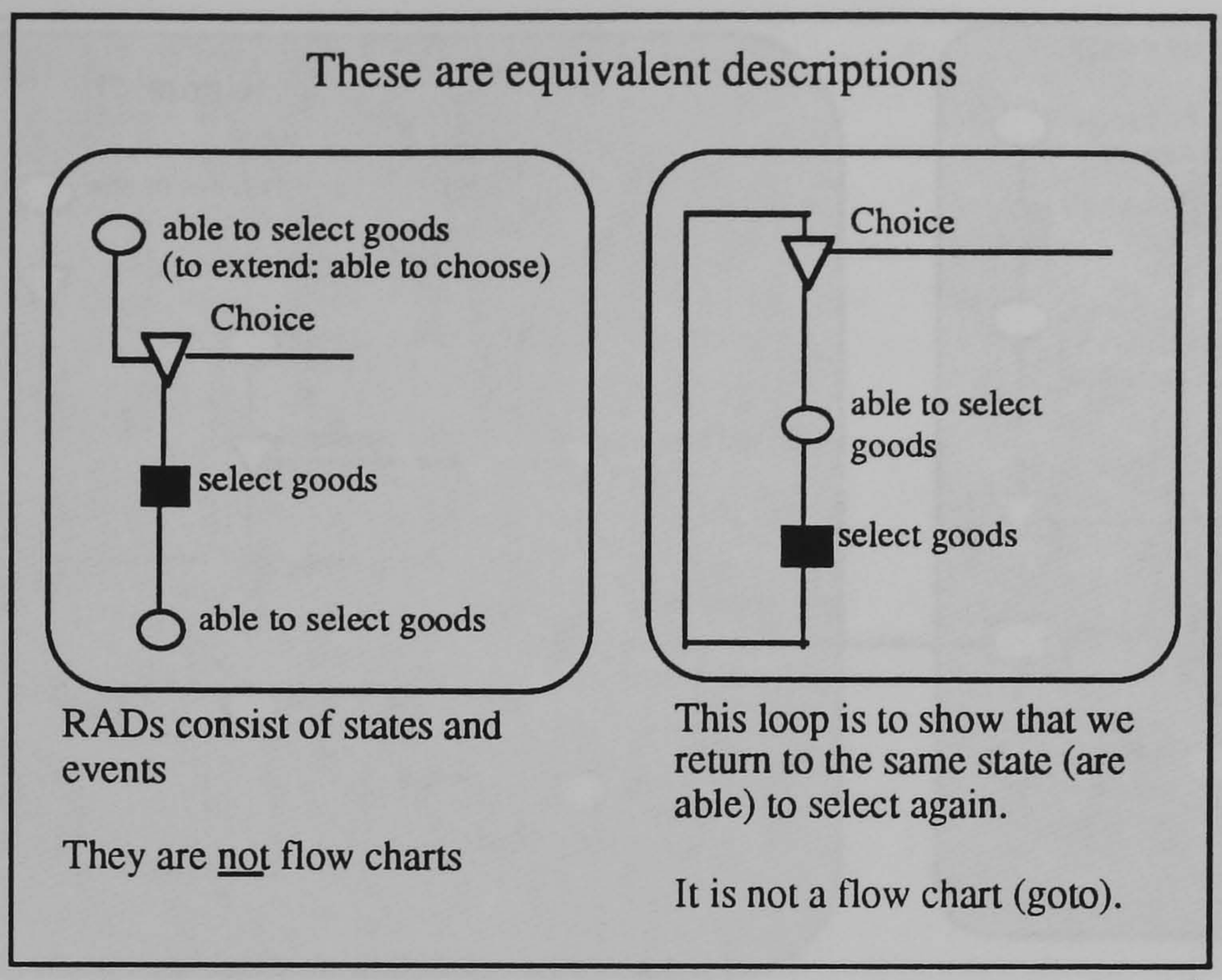
There are many approaches to providing notations for process modelling. We noted in the previous chapter that Madhavji's 1991 classification of modelling notations [74] needed 11 categories, and yet new notations are still being developed. Many of the notations within categories have subtle variations, yet often share a common view of how processes can be described. For example, IDEF0 diagrams, SADT and data flow diagrams though they have differences, essentially all depict the

process in terms of activities, inputs and outputs. It would be impractical to consider every conceivable modelling notation for a study (indeed the list may be growing too quickly). Hence, in order to choose a suitable process model we have chosen to consider three more common flavours, and to consider how they fit the modelling problem we wish to study. The three we will consider are RADs (Role Activity Diagrams: representative of role based approaches), CSP (Communicating Sequential Processes: representative of formal and mathematical approaches, and of approaches focusing on communication) and DFDs (Data Flow Diagrams are an activity and flow focused approach).(We will also give some consideration to variants of the data flow approach, since this is the choice that we made for our modelling). These approaches are very different in the way that they view process, and in their emphasis on what it is about a process which we need to know. We first give a brief description of each notation before considering their suitability for our problem.

3.2.1.1 Role Activity Diagrams.

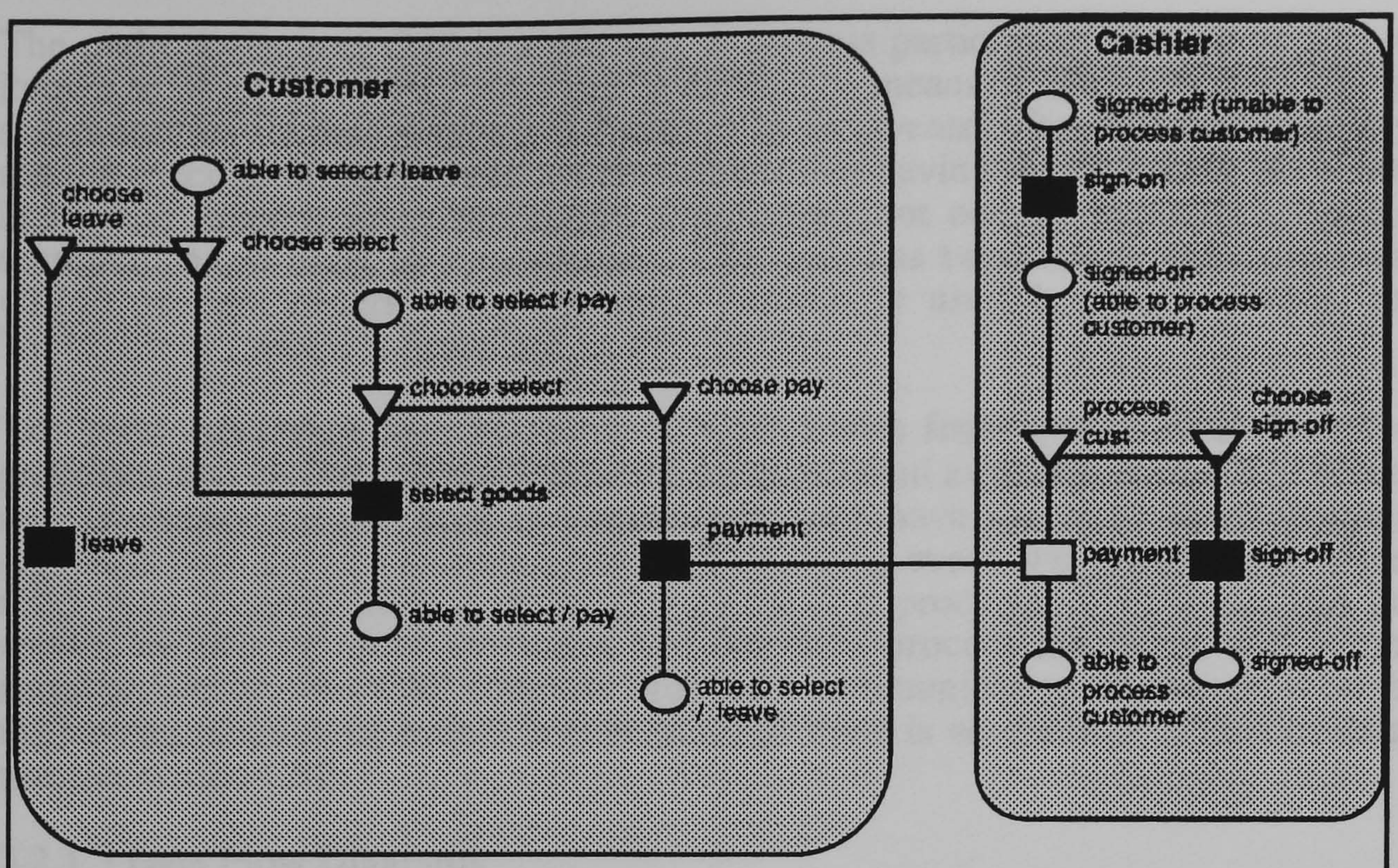
Role Activity Diagrams are a notation originally developed for software process modelling (from IPSE 2.5 work [117]). In the UK they have been used and promoted by both Praxis [36] and Co-Ordination Systems [33], and their merits have been discussed at a number of tutorials and meetings on process modelling - notably those supported by the IOPTClub [35]. A CASE tool for process modelling RADitor [231] marketed by Co-Ordination systems uses Role Activity Diagrams as its diagramming method, and a Role Activity Diagram front-end for ProcessWise Workbench (PWB) [232] is also under development. Role Activity Diagrams or RADs can be considered to be a state of the art single paradigm process modelling approach, and is well known among the process modelling community (particularly in the UK).

The central concept of Role Activity Diagrams is that of a role. A role describes a sequence of steps or activities which can be acted out by a person or perhaps by a group or department. These roles can be acted out in parallel and communicate through interactions (see below). It is important to realize that a role is merely a type. A single role can be acted by many people, and similarly a single person may have many roles. For example, one person may have a project manager role and an engineer role. Each role has a thread of activities (represented by square boxes) within it. The role is read from top to bottom, activities being connected by state-lines (the state between them). The intention is for the notation to be much more akin to Finite State Machine [209, 233] or to Petri-net [234] approaches than it is to flow charts, and some authors use a circle to label states in order to further emphasize this distinction [45, 231].



There are two kinds of activities within a role, actions and interactions. In Role Activity diagrams an action is a process step that the actor of the role carries out in isolation. Thus, actions do not involve any interaction with another role. An action changes the state of the role in which it occurs. Actions are represented by a shaded (we have shown as black) square on the Role Activity Diagram. An interaction between two roles infers that they have some shared behaviour, and is represented by joining activities (left unshaded) within different roles by a horizontal line. An interaction changes the state of the roles which are involved in that interaction.

Role Activity Diagrams also have two constructs for showing alternative or parallel paths within a role. Alternative paths are where the choice is dependent on some (yes-no) condition. This construct is usually denoted by an inverted triangle. However, often there are two or more independent actions which could be carried out at the same time or in parallel. These parallel vertical threads are denoted by the ordinary triangle symbol. There is no choice here, and thus no forcing down one thread, instead it is actually assumed that all paths are taken.



This diagram shows two roles: customer and cashier for a retail outlet, e.g. a supermarket.

Having entered the customer may choose to select goods or leave. Once goods have been selected the customer must make a payment before leaving. However, a number of selections can be made before paying.

On payment there is an interaction with the cashier. This is only possible if there is an instance of a signed-on cashier for the customer to interact with.

The main advantage of these diagrams are that they allow processes to be described from an abstract point of view. They allow us to show the roles and responsibilities of individuals and how they communicate with other roles. However, RADs make no assumptions as to the mechanism (implementation) of these communications. Hence, RADs are popular among the business process community, particularly those who are interested in re-engineering, because they allow processes to be described without focusing on what are considered constricting implementation details, like document passing mechanisms. Thus, the notation is able to enhance the possibility of more radical process re-design. Coulson-Thomas states:

'..the RAD diagramming technique is the most powerful way of representing the degrees of freedom or limits of empowerment of employees within an organization' [235].

3.2.1.2 CSP

CSP is a programming language based on concurrency and communication [236]. Though not originally executable Hoare's CSP has been implemented in a stepper [237] using the executable specification language Enact [238]. The stepper allows users to test the logic of the CSP by stepping through the permitted events. The stepper shows all permissible events and allows the user to select the next event (using either a command line or button-based interface). After an event is selected the user is again shown all permissible events, and we are able to carry on 'stepping' through the process and thus testing and experimenting with its sequences and dependencies.

The main concept of CSP is a process. A process participates in a set of events known as its alphabet. Processes can be defined by means of other processes (these sub-processes are then similarly defined) or by the events in its alphabet. Processes run (or execute) in parallel and are co-ordinated by having shared events. An event in CSP is instantaneous (an action which does not occupy any time), hence, communication must be synchronous. CSP also has two choice operators used within process descriptions; the most commonly used is simple choice, an 'exclusive or', which is described by the operator '|'.

The main advantage and flexibility of CSP comes from its ability to describe processes that can be executed in parallel (represented as parallel lines '||'). When two processes are executed in parallel and each have the same event in their alphabets they are said to share an event. This shared event, must occur at the same time, thus co-ordinating (synchronising) the two processes. The event can be shared by two or more processes and for each process the event will occur simultaneously; thus CSP supports broadcast communication. However, though the communication of events is synchronised there is no concept of data or data transfer within CSP.

3.2.1.3 Data Flow Diagrams

Data flow diagrams are a commonly used diagramming technique. They form part of a number of analysis methods, for example Yourdon [202], Schlaer and Mellor [239].

The key concept of data flow diagrams is the process (usually represented by either circles, bubbles, or rectangles). These processes can be decomposed hierarchically into other data flow diagrams. Each process is connected to either other processes or to stores (repositories of data). The connecting mechanism is the data flow arrow (hence the name).

The original data flow diagram does not convey sequence among its connected activities, although extensions to the notation have been developed in order to co-ordinate the 'firing' of processes [233]. Similarly some authors have also proposed executable data flow diagrams [240].

The main advantages of data flow are its ease of use, its readability, and its hierarchical process structure, which allows us to put together elements of process into one coherent whole.

IDEF0

IDEF0 also uses boxes and arrows to represent activities and flows. However, in addition IDEF0 has two further elements, mechanisms and controls. Mechanisms are those things which are used to perform the activity, normally people or machines. Controls are information which influence how the activity is performed. The IDEF0 technique is more complex to use and understand than data flow but does give a more rigourous description of the process.

3.2.1.4 Summary: Features of the modelling Techniques

Here we give a list of some of the features of these notations. Note that we have also included a final column for our later (TRADE) notation. (The TRADE notation is described in section 6.1).

Features	RAD	DFD	IDEF0	CSP	TRADE
Activity / Process	1	1	1	1	1
Co-ordination of activities	0	?	0	1	0
Controls on activity	0	0	1	0	0
Data sources / sinks	0	1	1	0	1
Ease of Use	1	1	?	0	1
Effort distribution	0	0	0	0	1
Executability	0	0	0	1	0
Hierarchy	0	1	1	1	1
Interactions	1	0	0	1	0
Mechanisms	0	0	1	0	0
Movement of Data	0	1	1	0	1
Order	1	1	1	1	1
Responsibility	1	0	0	0	0
Roles	1	0	0	0	0
Time-scale	0	0	0	0	1

3.2.2 Our Choice of Notation(s)

The choice of notation cannot be made simply by looking at how many features each notation has. Rather we must see how well the notation maps to our particular problems and context. We now describe how the choice of notation was guided for our particular study.

3.2.2.1 The GUIDE Framework

In order to help us choose modelling notations and strategy we have developed a simple framework (GUIDE: Goal, Use, Investment, Deliverables, Environment and experience) of things to consider for the modelling study. An example of this checklist (for the exploratory study) is given below.

Goal:	To understand (passive purpose) the launch process (object) at the site - from the view point (perspective) of the actors in that process.
Use:	Senior managers and other actors in the process (audience) will use the models in order to enhance their understanding (use 1) of the existing process, to aid discussion of it (use 2), and to suggest and communicate (use 3) improvements. The model will be used by a guide for enaction by people. There is no need for an enactable model (enaction).
Investment:	The initial (exploratory) modelling study is allowed only ten person days (effort). There will be no additional funding for a specialized process modelling tool. Interviews with staff will be limited to 1.5 to 2 hours each. There will be no additional time for staff training.
Deliverables:	Preliminary models of the documented and actual launch processes (d1). Report (d2) on discrepancies between documented and actual process. Presentation(d3) of key findings.

Environment:	Existing procedures focus on activities and products. The engineers and managers are comfortable with procedural notations.

The goal of the study was our main starting point. Having decided upon understanding the launch process as the first goal (as above) we initially considered adapting the GQM paradigm [174] in order to aid the model notation selection. This would then have a three stage hierarchy from goals through questions to notations, i.e. GQN: Goals, Questions, Notations (see below).

Goal: To understand (passive purpose) the launch process (object) at Site X (environment) from the view point (perspective) of the actors in that process.

Q1: What are the activities in the process?
N1: Data flow diagrams
N2: MVP
N3: HFSP

Q2: Who is responsible for project initiation?
N4: Petri-nets
N5: Role activity diagrams

It should be apparent that certain notations could be attached to questions. For example, question 1 can be linked to notations N1, N2 and N3, whereas question 2 is much more likely to link to notations N4 and N5.

However, though this method might lead to the notation which was finally adopted, we believe that it does not sufficiently stress the importance of a number of factors. Hence, though we have kept the goal as the major factor (and kept the structure of the goal exactly as in GQM) we have further emphasized other factors. For example, though the environment and perspective are noted in the goal definition, the effect of the experience and priorities of users, and the way in which the models will be used may be overlooked. In addition, the amount of investment to be made, and the resources available for the study need to be taken into account. We now consider these factors in greater detail, in relation to each study.

3.2.2.2 Exploratory Work

The intended audience for the models (use) was senior management and actors in the launch process. It was important that both of these groups of people would be able to understand and validate the models produced. In addition, the limited amount of allocated staff time (investment) meant that we were not able to take time to train people in the use of a notation. Hence, we needed our notation to be extremely easy to use, and readable. We knew that we wanted to focus on both the existing actual and theoretical processes (deliverables). It was clear that the existing procedures (environment) used much terminology which would map well to an activity and flow based notation, because there was a focus on activities producing documentary products.

Both IDEF0 and dataflow would have been suitable techniques. However, there were few descriptions of process controls within the existing process documentation. Hence, we would get little benefit from this added IDEF0 feature. In addition, we felt that the IDEF0 notation was slightly harder to learn and understand. Coulson-Thomas states that in order to use IDEF0:

'Both the analysts expected to undertake the modelling and the individuals within the business require significant training in the techniques for meaningful models to be built and communicated' [235].

The constraints on the amount of time we were able to spend with staff at the site meant that training them in IDEF0 was not feasible. In addition, we did not at this stage consider examining resource (part of the IDEF0 mechanisms). So we chose to adopt data flow diagrams as the modelling approach. An added bonus of this choice was that we had access to a suitable CASE tool, some experience of using the notation, and access to other data flow 'experts' who could be used to check and validate our models. However, we note that this is a small distinction, and that although there are differences between these two notations they adopt essentially the same view of process.

Data flow techniques have been criticised as process modelling notations for their focus on process implementation detail, such as document passing [235]. Such detail can make it difficult for modellers to break links with the previous process when attempting to re-design. However, we were not interested in using process modelling in such a revolutionary (re-engineering) way. Our primary concern was with existing process detail and how it differed between theoretical and actual process. Hence, the choice was an appropriate one, and it did enable us to discover discrepancies between the actual and theoretical processes.

3.2.2.3 Later (Instance) Work

Again we adopted a goal based approach to deciding upon our modelling notation. For example, we started by having a goal of investigating the relationship between process and project success. One of the key questions that this had led to was "how do we characterize software project launches". Thus, we decided that in examining instances of process, it was necessary to be able to characterize or describe pertinent aspects of projects, and judge the success of projects. This can again be seen as a GQM style hierarchy from goal through questions, to the model or measure characteristics that we needed (see below).

Goal	To investigate the relationship between launch process and project success at our collaborating site.
Question 1	How do we characterize or describe pertinent aspects of projects?
Question 2	How do we judge the success of projects?
Model / Measure for Q1	Need to model activities and data in project launch. Need to measure effort expended on activities.
Model / Measure for Q2	Success scores (see description in Chapter Five).

We did not have an existing modelling paradigm, which would allow us to satisfy the requirements of our study. For our initial study we did not wish to focus on resource, and thus standard data flow diagrams had been adequate. However, this later work which did examine resource had a different more quantitative point of view. In this work we wanted to specifically look at the effort expended by people. In addition, rather than have this merely as an input to an activity (as in the IDEF0 mechanism) we wanted the model to be able to show this dimension graphically. Hence, the main interest for the study was on activities, and the pattern of resource usage (human effort) among those activities.

We will discuss the judging of project success in Chapter Five. However, the need to collect and model both launch activities and effort expended on those activities led us to develop the extended data flow notation TRADE. (This notation will be described in Chapter Six - section 6.1 introduces the elements of the notation and the project models produced are shown in section 6.2.2.2). An additional benefit of extending the notation (DFD) which we had already used was that it allowed us to present our models to managers with only a minute or two of explanation. (Once again we had no time allocated for training). We will further examine our case study design for this later work in Chapter Five, however, we show below our GUIDE framework which again helped us to choose a suitable notation for the study.

Goal:	To investigate the relationship between launch process and project success at our collaborating site.
Use:	Project managers (audience) will use the models in order to provide insights about their projects (use 1) and to identify key project activities (use 2). The models and findings will be also be used and to highlight, discuss and guide successful practices (use 3).
Investment:	Access to five project managers. Maximum for interviews of 1.5 to 2 hours + follow up of 30 min. to an hour. There will be no additional time for staff training.
Deliverables:	Depiction of the launch processes of the five projects to be studied (d1). Analysis of results to attempt to discover key process activities (d2). Mechanism to facilitate the data collection (d3)
Environment:	Engineers and project managers are comfortable with data flow notation used by previous study.

3.3 Summary

We have presented the reasons for our choice of the case study as our research method, and our choice of notation for our exploratory and later (instance) studies. (We chose to name our later examination of projects instance studies, since these concentrate on specific instances of the process). The following two chapters will describe the planning, design, conduct, and use of these case studies. Specifically Chapter Four examines the exploratory case study and Chapter Five the instance case - which examined five software projects. In presenting these coming chapters we have been guided in our framework by two main sources; the pilot study framework presented by Glass [241], and the DESMET case study design and analysis module [228].

4. The Exploratory Case Study

Chapter Synopsis

This chapter describes the planning, design, conduct, and use of the exploratory case study. In addition, we include a brief critique of the work carried out.

4.1 Exploratory Case Study Planning

Rather than being purely an investigation of process modelling this work addresses a specific process problem; notably the dissatisfaction with the current launch process.

This can be decomposed into two given sub-problems:

1) Lack of understanding of the launch process
2) Users deviating from the launch process

and into related questions.

1) About the process

a) Is launch process a suitable candidate for study?
b) Do the users really know what the standard process is?
Is deviation by ignorance or by design?
Does it matter?

Initially we wanted to discover whether there was deviation from the theoretical process (as was suspected by the process designers). Although ideally we would wish to do this for a number of projects, and compare projects, interviewing a number of people on each, the limited staff availability did not allow this. We could have either interviewed a number of people in one project, or across projects. We chose the latter of these so that we could get a wider view of the process, for less effort. The plan was to show from this generic view that there was deviation among projects, and then (having shown this) to examine individual projects in the later work.

Therefore, our initial exploratory study, concentrates upon looking across projects and showing what people really do, and differentiating this from the theoretical process (what the organization assumes that they should do). In addition, the interviews attempt to find out whether deviation from process is always intentional or whether it occurs through a lack of understanding.

Success criteria

(a) Identifying discrepancies between procedures and process.
This will show that the launch process is a suitable area of process for further study, since it will show that process deviation occurs.
(b) Showing where the process is mis-understood.

2) About the exploratory process modelling

a) Can process modelling with simple notations help us to increase understanding of the launch process?
b) Is the notation used (DFDs) a viable choice?
c) Is the strategy adopted a viable choice?

Success criteria

a) Identifying discrepancies between procedures and process.
b) Finding or identifying problems with the existing process. Showing that the notation has allowed us to uncover what we wished. That it is an appropriate choice.
c) Being able to convince the organization of the worth of the study, and of further study.

4.2 Exploratory Case Study Design

4.2.1 Sources of Information

Owing to the restriction in access to staff time the tasks and schedule for the exploratory study have been agreed over a series of meetings with the site quality manager and engineering director. Our main sources of information for this study will be documents (both describing, and produced by the process) and interviews with process actors.

For examination of both the theoretical and actual process we wish to include not only the use of the launch process within engineering, but also its major interactions with other processes. Therefore, in studying documentation we will include study of the procedures which have and impact upon project launch. For interviews we have limited access to staff, and have decided to limit ourselves to interviewing staff from three sources, namely engineering, marketing, and the separate project support function.

In order to model the theoretical process we will examine two main sources of documentation: procedures documents and templates for the production of launch process documents. In each case we will produce separate models at the end of each phase (see our comments on phased modelling in sections 4.3.1 and 4.3.2)

For examination of the actual process we will examine documents produced by the process, before moving on to interviews. After each interview, we will update both models and questions, in order to make more efficient use of later interviews. We will finally have a further round of interviews to validate the users' understanding and agreement to our models. Thus, the modelling will have five distinct phases:

Theoretical process:
1) Examine procedures
2) Examine document templates

Actual Process:
3) Examine previous documents produced
4) Interviews (re-model and restructure questions after each).
5) Discussion of models produced with staff.

4.2.2 Evaluation of Results

At the end of the study we will present our findings to the organization. This will take two main forms

1) Report: An internal site report. This will be drafted and then revised after feedback from the quality manager.
2) Presentation and Feedback: To all those involved in the study and any other interested parties, including the engineering, director, quality manager, and marketing and finance representatives.

These two reports will attempt to present the discrepancies and process problems (as outlined in our success criteria) which we have uncovered by utilizing our chosen modelling paradigm. For our own aims a successful outcome of the initial exploratory case will be the go-ahead for further work. However, a further vindication of the worth of work would be if the organization decide to revise the launch process procedures documents as a result of our findings.

4.2.3 Verification and Validation

One of the phases of the exploratory study will be to meet with each process user interviewed to discuss whether the model produced is a faithful representation of their process. We can at least know that the models are useful, unambiguous and understandable if users can point out areas which they believe are incorrect, and that can consequently be modified.

However, we cannot say that these are the only possible representations of the process using the particular technique. One of the problems in attempting to validate the modelling effort is that this is to be carried out by a single individual, and thus we cannot have comparison of two models of some process aspect. (We have no further resources to be used in this kind of validation effort). In addition, analysis notations like Yourdon, leave the modeller with a fair degree of flexibility (even compared to methods like SSADM). Hence, the modeller will make some choices in representing process, as a result of process study and interview. However, the process of reaching agreement with process users, and of producing models which (by consensus with a number of staff at the site) are felt to be representative of the launch process will help to show that such choices are reasonable and not misleading.

4.2.4 Benefits

We are not able to provide quantitative cost data for the benefits of increased understanding of the launch process. This study has not concentrated on evaluation or on investigation of specific projects, and thus we cannot make statements of the kind 'process improvement has led to a saving of ..'. Consequently benefits are seen in the attitudes and feedback of process users, and in any subsequent process change.

However, it is considered [242] that mistakes made early in the process (in requirements and specification) incur extremely large costs to projects. Hence, any clarification of this process is potentially a significant cost saving. In order to investigate this, we would need to examine and track a large number of projects from the launch phase to delivery and maintenance. However, this is not within the scope or resource of this research.

4.2.5 Deliverables

We have agreed to the following deliverables:

1) Preliminary models of the documented and actual launch processes.

2) A report detailing discrepancies between the documented and actual versions of the launch process and suggesting possible improvements.

3) A presentation to senior managers, process designers and process users to report back key findings.

There is to be no quantitative study or project evaluation in this exploratory case.

4.2.6 Questions for the Case Design

Here we use the design questions section of the DESMET Case Study Design and Analysis (CSDA) Procedures [228] to consider issues pertinent to the exploratory study. In order to gain the widest view of process possible with the minimum impact on the organization, a number of staff will be interviewed across projects, taking in their experience of different projects.

Q1 Baseline to compare the results of the evaluation.

Projects were not given a treatment. Projects will not be examined individually. The modelling here is post-project, in order to see if we can discover insights about the development process. That is we wish to discover whether the process modelling increases understanding of the launch process. Hence, our baseline is the existing process description.

Q2 Constraints

Access to information for the modeller.
Time-scales for the modeller.
Impact of interview time to organization personnel.

Q3 Hypothesis

Process modelling using simple notations can increase understanding of the launch phase of the development process.

$Treatment_0$:	Baseline. Existing process description.
$Treatment_1$:	Process modelling using data flow diagrams
Response variables:	Discrepancies and problems with existing process. Qualitative. Discussions and feedback from process users about the level of understanding with the existing process description and with the data flow models.
State Variables:	Process modelling (using DFD) applied or not applied.
Effect Investigating	Understanding or insight.

If we are able to demonstrate that the above hypothesis is true, then this will satisfy the success criteria for our questions about process modelling (question 2 in section 4.1). That is, it will show that the use of the simple modelling notation (DFDs) and the strategy (phased modelling) were viable choices which allowed us to uncover process problems and discrepancies,

and increase process understanding. Furthermore, this will also show that the choice of process area was an appropriate one, and that we are able to illustrate areas of process which are mis-understood, thus satisfying our success criteria for questions about the process (question 1 in section 4.1).

Q4 What are the response variables?

We are really investigating something (understanding and insight) which is measured as a perception. Indications of this are problems found by using the modelling notation, and the feedback which we get from process users.

Q5 What are the experimental objects?

Project(s) In order to gain the widest view of process possible with the minimum impact on the organization, a number of staff to be interviewed across projects, and taking in their experience of different projects.

Q6 When in the process will the modelling take place?

Although we are using process modelling to describe the launch phase of the development process, the models for the exploratory study are based on experiences of past and present projects. Hence, there is no single process phase in which the modelling takes place.

Q7 When are measurements of the response variables taken?

As noted in questions three and four, our response variables are discrepancies and process problems found by the modelling and the feedback gained from process users. The majority of this feedback will occur during the presentation (to the organization) of the process modelling findings.

Q8 What data will be used to allow measures to be calculated?

We calculate no measures to support this initial study.

Q9 Can the effects of the treatment be isolated (from any confounding factors)?

Typical confounding factors to consider at the site are the allocation of staff to projects and the time at which the method is applied.

Staff

Staff were already allocated to projects as normal in the organization, our case study has no control over this. However, we will compensate for this by interviewing a number of staff across projects.

Time

The modeller was using the method over the same period of time. Therefore, there is no learning effect on site staff to bias the results.

Q10 What are the procedures to ensure that the method / tool is used correctly?

There is some sacrifice of absolute terminology (Yourdon) rules in order to fit in with organizational terminology, e.g. use of the term concept to imply the activities of the concept phase.

However, all production of models is by one person who has significant experience (over 2 years) of using the notation.

Q11 To what extent will the treatment be integrated into the current process?

There is no intention to integrate the exploratory modelling into the process.

Q12 What are the state variables / project characteristics which are important to the study?

The initial exploratory work looks across all projects, and does not try to measure or control project characteristics. All projects have the same application domain and should follow the same (launch) process.

Application area:	Software for inclusion in board test equipment. Must be applicable to these.
Methods / tools:	No imposed development method (e.g. Yourdon) at the site.
Development process:	All projects should follow the same (launch) process under investigation.

However, the state variable is the use (or not) of process modelling (see our hypothesis in Q3). That is we are investigating what happens when we apply process modelling to the launch process at the study site.

Q13 To what extent do we need to be able to generalize the results?

Our results need to be applicable only to the study site. The intention is to use the exploratory study to demonstrate the potential for further process modelling work at the site. The exploratory study may then be used as a baseline for this later work .

Q14 Confidence in evaluation results

We are looking for qualitative evidence that the process modelling technique can provide insights into the launch process and improve process understanding. This exploratory study, is only a feasibility study. Our aim is to show process users that this is a potentially powerful technique, and to persuade them that further work will be beneficial.

Q15 How to analyze the results of the evaluation

If we are able to use our modelling technique to identify and demonstrate process problems, and to increase process understanding, and if this is considered to be of worth to the organization (by them) then we will consider that the study has been a success. (Our evaluation of this is given in section 4.4). This will satisfy our success criteria (section 4.1) and our hypothesis (Q3) that process modelling using simple notations (in this case DFDs) can increase process understanding.

Q16 Appropriate level of confidence?

As we are only testing the feasibility of our approach, in order to argue for further study, confidence in the results is not a major factor in the exploratory study.

4.3 Exploratory Case Study Conduct

4.3.1 Modelling Strategy or Framework

We have described above how modelling (i.e. understanding the process) and producing the models of the process involved five distinct modelling phases. These activities can be seen as part of a larger exploratory case framework, namely:

1) Understanding the theoretical process:

1.1) Examine procedures.
1.2) Examine document templates.

2) Understanding the actual Process:

2.1) Examine previous documents produced.
2.2) Interviews (re-model and restructure questions after each).
2.3) Discussion of models produced with staff.

3) Noting discrepancies between the actual and theoretical process

Undertaken throughout the modelling activities. To be collated as an internal report and as a presentation. (See use of the models below).

4) Understanding the process and showing areas which can be improved.

Again this forms the major part of the report back to the organization.

5) Making process changes based upon findings.

Hence, there were two major modelling tasks: to develop a model of the intended or documented process and to build models of the actual process. Descriptions of the 'intended' or 'theoretical' process could be found in two distinct types of document. The first type is what would be commonly thought of as procedures; mainly text with some diagrams, usually flow charts. There were also other relevant types of these 'procedures', general procedures (company wide), engineering procedures, and local procedures. Though all of the specific launch process documents were cross referenced, the links to other procedures documents, which often affected process launch, were not always clear. In addition, to procedures another form of process description (if more implicit) were the templates for the documents to be produced by the launch process. These were available on disk and included substantial amounts of hidden text guidance[28] for the template user. Templates were provided for each key document in the launch process, and were always used, though some admitted to not using or ignoring the hidden text guidance.

The aim of the exploratory case was to produce a final general model of the documented process and this was achieved by five distinct steps. 1) Separate models were produced based upon procedures, and then 2) procedures and

---

[28] Hidden text is text which can either be activated so that it appears on screen, or in the printed document, or switched off, at the discretion of the user. Thus, it can give guidance when carrying out the task of completing a document, without appearing in the printed version of that document.

templates. We then 3) studied real documents output by the process, 4) interviewed a representative selection of actors in the process, and then finally 5) we validated models produced in a second round of interviews.

At each stage or phase the apparent contradictions, gaps in understanding, and so on were noted both on the model and in an accompanying list. For example, we found activities which produced some product which then apparently went nowhere, and which did not appear to be used by any other activity. Similarly we found that a product might be mentioned for which we could find no activity to create it. These kinds of issues were made very clear by the data flow notation, because unconnected flows, or missing activities were easily visible. These kinds of problems or discrepancies, along with holes in understanding, were then used to help direct enquiry at the next stage. This technique was particularly useful when it came to interviewing users of the process, because it allowed questions to be formulated, both before commencement of this phase, and after each subsequent interview. Finally we attempted to contrast two models, one based upon the process description (procedures and templates), and one based upon the actual process (mainly from interviews, but also some information from documents produced by the launch process).

We considered it important to study documentation prior to interviewing process designers and actors for three reasons:

1) Influence: We feel that a modeller will be less swayed by initially looking at documents, than by talking to potentially persuasive process actors[29].

2) Credibility: The modeller must be seen to be worth talking to, and must be confident in his or her ability to interview actors in the process. This is highly unlikely without the modeller having some prior knowledge of the process in question.

3) Efficiency: The interviewees are invariably busy people. It is far more efficient to have already examined the process to some extent, such that questions can be targetted to some degree. It is not wise to waste the time of those who you have persuaded to talk to you[30].

4.3.2 Modelling Issues, Thoughts and Issue Resolutions

The main way in which we tried to understand the process was to attempt to describe it with our chosen notation; thus our modelling was not simply an output of the study, but also the primary way of gaining understanding. The phased modelling method relied upon the modeller only using the material suggested by each successive stage in order to create the model. In effect, the phases map to an increasing level of understanding, on the part of the modeller. Many of the

---

[29] An example of this is that the iterative and phase like nature of the process; in which level of detail is increased in successive iterations of the product proposal document; was picked up from procedures documents. However, subsequent interviews persuaded the modeller that this was not the case, and it was not until phase one models were re-examined that this error was discovered. People are very persuasive!

[30] We found that access to personnel can be a major problem in such a study. In our case we were constrained by time (from the modellers point of view), and although we received excellent co-operation from all those who we approached, the demands on their time sometimes meant that they were genuinely unavailable.

ambiguities found or questions raised by earlier phases are answered in modelling later ones. However, keeping explicit models of each stage means that these problems or gaps in knowledge at one stage are not lost or forgotten when such further enquiry (at a later stage) leads to greater understanding. Although we finally presented differences at a coarse grain (between documented and actual process), having kept the phased information (models) allowed us to much more easily justify our arguments. For example, we were able to make a point about the documented process, show it on a procedures model, and even relate this back to the relevant section in the document - and this made it much for easier for staff to decide where changes needed to be made.

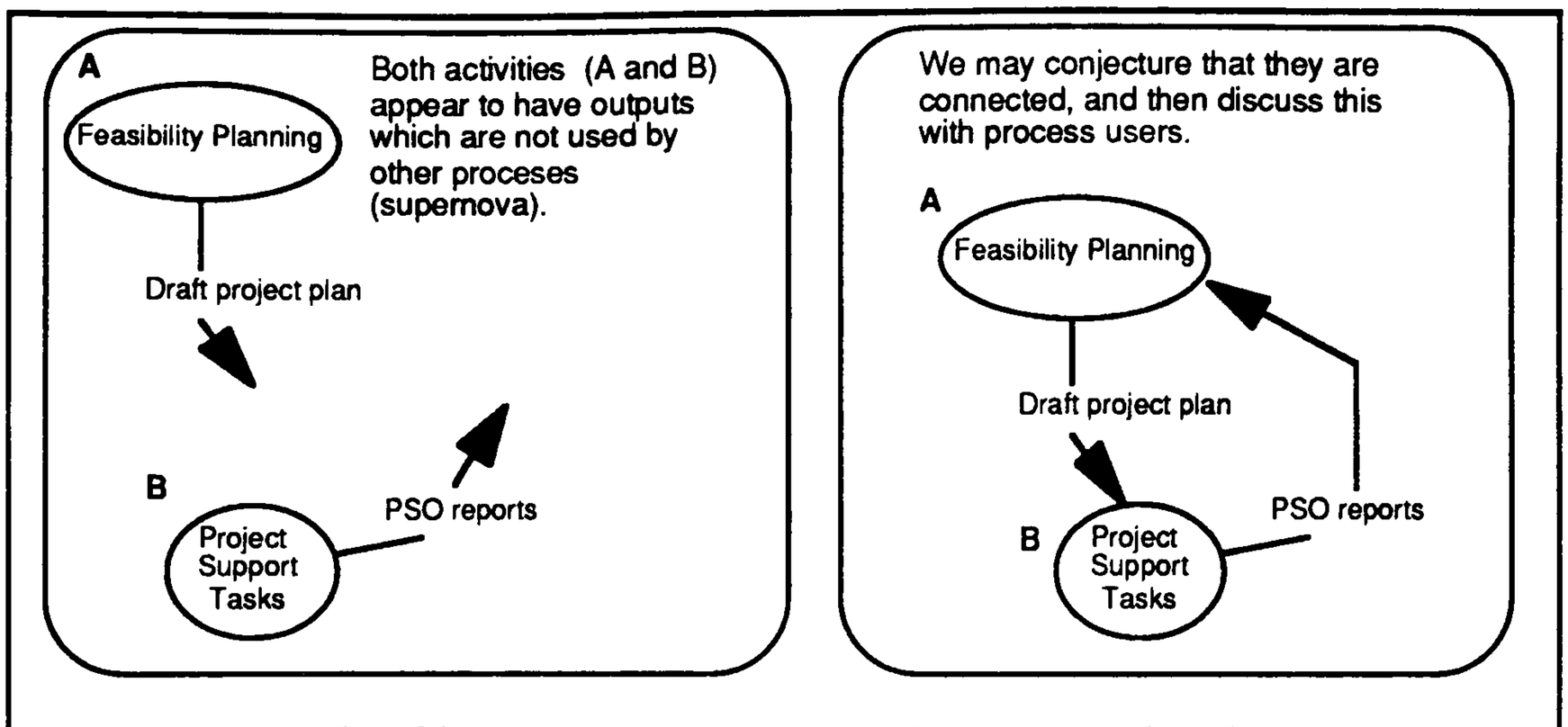
A problem which has been touched upon in this section is that the view of the real process is coloured by having already modelled procedures. This points out an apparent dichotomy between using modelling as a learning exercise and using modelling to create a particular view of the process. One solution would be to use different people to create the various models. However, this presents difficulties of its own. Aside from the overhead, it has been found by this study that it is only by having some idea of the process that effective questioning could take place. Though not having fully resolved this question, it is felt that the discrepancies between phases which are found will be no less valid, and we hope to have erred on the side of pragmatism.

4.3.3 Mechanism for Using the Models

We have stated that the data flow models were used in order to facilitate process understanding. However, we have not described a mechanism for comparison between models, or for identifying process problems from the models (Note that all of the original (unaltered) data flow models produced by the exploratory study are included in Appendix A).

Identifying Problems

In order to discover process problems we simply used existing data flow or Yourdon analysis heuristics to uncover weaknesses in the model. For example, we would start by producing a model which pedantically described procedures. We would then look for what are termed supernova (activities or stores with output and no input), black holes (activities with and stores with only input), dangling flows (which leave an activity and then go nowhere or which come into an activity from nowhere). We would also make extensions to models, based on conjecture of what the process might be supposed to describe, and on interviews, and then use these extended models to ask people whether this represented their process.



An example of how we may use the models to examine the process.

It is significant that the majority of problems were uncovered using these kind of simple techniques, and that no automated analysis (e.g. by transposing the model to some executable notation and then running simulations) was used. Indeed, this fits with the use of the notation for systems analysis, and we felt that no more formal effort was necessary.

Comparing Models

All comparison of models was entirely subjective. We were essentially looking at differences in patterns, since conflicting terminology had been resolved by this stage (typically by the interviews and validation interviews). However, it is important to realize that this often involved lengthy discussion with staff at the site. All reporting of the differences between views (e.g. theoretical and actual models) was described and commented on both in the internal report and the presentation at the end of the study period.

4.4 Evaluation of the Exploratory Case Study

4.4.1 Successes

4.4.1.1 The Process Models

The process models proved useful in:

1) Making process knowledge visible.

The modelling activity not only aided the understanding of the modeller, but by presenting models to users allowed validation and discussion using a common framework. We found that we could check our understanding either by asking specific questions (we used models to say "do you mean this?") or by allowing people to point out on the model, or annotate it, where they believed it to be incorrect. This enabled us to produce a description which more closely followed the real process.

2) Providing a basis for discussion.

Presentation of our view of the process, and our findings promoted a great deal of discussion not only about process problems, but also about possible

solutions. We believe that this is for two main reasons. Firstly, the models provided a succinct, compact and understandable view of process, and could be used to illustrate ideas about process change. Secondly, we believe that an independent or external view of the process was appreciated as worthwhile, and prompted people to re-evaluate their perception of the process and their own role within it.

3) Giving a coherence to procedures.

One of the major weaknesses, particularly in the current procedures appeared to be a lack of integration. The links among various process areas were often unclear, and the documents did not all have the same format and terminology. Graphical models of the process not only highlighted the lack of coherence, but were used to identify solutions.

4.4.4.2 The Study in General

The approach of starting small and aiming for some small success paid dividends. The study was seen as successful by the collaborating organization, and it paved the way for further work. In addition, it had intangible benefits, in that it helped to increase discussion about process, and a feeling of process ownership among participants. The presentation particularly, proved to be a lively forum for discussion, with many comments on the models, ideas put forward and suggestions for further enquiry. Another significant factor was that those who had been involved felt quite positive about their contributions, for example, one project manager actually said "I think you've summed up the problems we have here". These people became allies in attempting further process modelling work, and were happy not only to be approached about further participation, but to approach others on our behalf. Finally, the study increased our visibility within the organization as a whole, not just within the engineering function. This generated interest in notation, in ways of modelling procedures and so on. More importantly it meant that we got real (directorate) backing for further work, and that there was much more confidence in us, and co-operation from future participants.

4.4.2 Failures

The biggest problem with the initial case is that it did not provide any insight into the effect of process upon project. This was perhaps inevitable given that we did not look at individual projects separately. However, even such a study would really need some attempt to measure such effects (which we tried for our later study).

In addition, we have no evidence (other than apocryphal testimony - see above) to show that process perception or process understanding was actually aided by the study. Even a before and after subjective score from individuals involved could be interesting. However, there was clearly a feeling that the study had been useful. It is significant that actual process changes (to procedures and to templates) were made as a result of the study.

4.4.3 Cost benefits for future usage?

The amount of time used by the interviews of site personnel, was approximately one person day in total. We estimate that including study of documents and production of models the study took approximately ten person days. Despite the modest effort used by this exploratory study the work has resulted in significant benefits to the organisation. We are unable to quantify the effects of such benefits because we have no measures of project improvements.

However, we conjecture that if process understanding can be improved across projects, the benefits will outweigh the cost. Increased process clarity will aid early understanding of the project. This understanding will help to detect (or prevent) errors much earlier than they might otherwise have been found. Errors found late in the project (e.g. in testing) may be 100 times more costly than those found early (in the launch phase) [242] . Indeed, we found one project at the study site in which a single launch phase decision accounts for a protracted and expensive testing period, that far exceeds the amount of time spent on our study. Thus, the total cost of our study could be repaid even if it only helps to prevent or detect a single error.

Furthermore, for our later study of individual projects we will use a strategy which requires far less effort for each project so that ultimately there should be a cost benefit for each project studied.

4.4.4 Recommendations for future use

4.4.4.1 Critique of the exploratory case

The exploratory case was a success: it produced its deliverables, it provided insight into the problems of the launch process, and it paved the way for further work. However, it also showed that we would need to refine our modelling methods in order to proceed with the examination of individual projects. (We discuss these issues in the following section).

More generally, the way the information had been gathered and the interviews conducted appeared successful, and as a result the organization were happy to grant further access to data and staff, many of whom had made it known that they would be happy to take part in further work of a similar nature.

4.4.4.2 Lessons Learned

In addition, to answering the questions raised in the case study design (and testing the hypothesis) one of the less tangible successes of the study was that it satisfied our aim of obtaining experience of process modelling in practice. From this experience we have compiled a list of lessons learned, to be used by similar future studies, which is given below.

1) Concentrate on the goals and characteristics of the organization.

   Start by letting the organization suggest the problem or opportunity to be modelled. This means that the modelling effort is viewed much more positively and modellers are seen as people who can help. This much more likely to result in the necessary co-operation.

   Most of our study was to be carried out within the engineering function at the site, and thus we agreed on our goals with the engineering director and the quality manager. However, the presentation of findings took in a wider audience so that other business functions were represented.

   Take into account the environment, actors and potential users. It's no good having a wonderful model that no-one will understand, or be able to use. We aimed the models at being understandable to process users (engineers and project managers) and to other managers (often without software engineering backgrounds).

2) Choose a feasible goal for the process modelling study.

It is better to do something simple at first which succeeds, than to just fail to deliver a wonderful but complex piece of work.

3) Have a champion of 'your cause' (the modelling study) within the organization.

It is much easier to get things done if you have support from a senior — preferably Board level — manager.

4) Don't criticize any individuals. Simply assess the process.

As part of our study we needed to uncover what people really did. Sometimes these actions would be helpful to their software project, and sometimes detrimental. However, rather than be judgmental we took the stance that deviation from the procedures might be a good thing, and that process evolution might recognize or include this deviation. However, if we criticised individuals for taking such action, then there was far less chance of them telling us about it. Instead they would be much more likely to tell us what they thought they should be doing if they followed procedures.

5) It is a huge benefit to be seen as independent.

People will tell you things that they would not tell their managers. However, if you are perceived as doing this for Mr X, then they may be more reluctant to do so. Note that you must also try to safeguard confidentiality.

6) Explain what you are doing and why.

This is particularly fruitful if there is a perceived problem, which the modelling work is attempting to address. For example, we found that everybody had at least one gripe about the current process. If they think that by telling you things may change for the better, they are much more likely to take an interest.

7) Be honest. Confess your ignorance.

People usually like to help. As an external observer you are not likely to understand the problem domain as fully as the process actors (despite having devoured procedures manuals, etc.). Ignorance is one of the reasons that you are interviewing staff.

8) Make the organization take decisions.

Present the facts, and let them decide what the implications are, and what they should do. In other words have a supporting or facilitating role.

9) Get process users involved in discussion about the process.

This is one of the benefits of having a process model to distribute for comment. It is opportunity to give a sense of process ownership which should not be lost.

10) Be prepared to be flexible.

Many people have tremendous demand on their time. You may not always be the highest priority.

It is interesting to note that other authors who have carried out programs to introduce new technologies within an industrial environment also give similar advice. The most striking similarity is with Pfleeger's work at Contel [181], which also gives ten 'lessons learned'. It is not surprising that some of our conclusions are the same, indeed this study actually used Pfleeger's advice about starting small in deciding that a smaller exploratory case (pilot) study was an appropriate strategy. However, the fact that other lessons learned were confirmed by this work is remarkable, notably that one 'should start with people who need help', and 'use different strokes for different folks' which are equivalent to our point one, and 'criticize the process and the product, not the people' which is equivalent to point four. Indeed, such lessons learned though they may appear to be obvious (or common sense to some) can be extremely valuable, and are the kind of advice that is often so lacking for the potential practitioner. We believe that once again this points out and confirms the merit of industrially-based case study work in software engineering, particularly when introducing relatively new technologies.

4.5 Use of the Exploratory Case Study

We have found that a simple modelling notation can be utilized in order to provide insights about some aspects of the development process (in our case the launch process). However, before moving on further study we need to re-assess our approach, in the light of the exploratory case.

By providing some success the initial study suggested to us that we were 'on the right track'. The data flow models had been well received as had the insights which the modelling had provided. Furthermore, all agreed that there was a need to look at the same process area in greater detail. However, the lessons of the exploratory study also suggested some changes to our subsequent strategy, so that differences between projects, and between projects and procedures could be investigated.

1) We needed to move towards reference and instance models.

   We needed to produce separate models of individual projects in order to see whether process deviation was a problem. Only by comparing separate projects with procedures, or with each other, could we assess the impact of process deviation. In addition, we took the opportunity to remodel the procedures documents in order to have a baseline model for future change.

2) We needed to collect effort data.

   Projects which have the same or similar activities may in fact be very different. For example, similar data flow models may have very different distributions of effort among those activities. Furthermore, since much of the point of doing our modelling exercise was to determine what were worthwhile launch activities we decided to incorporate effort data in our later study.

3) We no longer needed hierarchical models.

   For the initial exploratory modelling we had been keen to use a hierarchical notation, so that we could incorporate various levels of process detail into some coherent structure. One reason for wishing to do this is that we did not have an idea of what level of detail we wished to focus upon. However, this work had allowed us to find that the process deviation occurred at the level of production of revisions of product proposals and planning

documents. Hence, we decided that the lowest level of detail we would concentrate upon was the production of a single revision of these documents, and that we would also collect effort data at this level. This meant that we could abandon hierarchy in favour of flat models, at the same time showing a true project time-scale. (See our discussion of the TRADE: Time-scale, Resource, Activity, Duration, Effort notation in Chapter One (section 1.5) and in Chapter Six (section 6.1)).

5. The Instance Cases

Chapter Synopsis

This chapter describes the planning, design, conduct, and use of the instance case study. In addition, we include a brief critique of the work carried out.

5.1 Instance Case Study Planning

Although the exploratory study has produced models of actual process, these are based on a mishmash of several projects. Interesting questions centre on how specific projects deviate from process descriptions, and from each other. Only by examining projects, and producing project models, are we able to examine the relationship between process and project success. This raises a number of questions:

1) About the process

   a) How are some projects different from each other?
   b) Does process affect project success? If so what is it (in process terms) that some projects do that makes them more or less successful than others?
   c) What are the key project launch characteristics?
   d) How do we describe these aspects of projects?
   e) How do we judge the success of projects?
   f) How can we relate the success of projects to launch characteristics?

For each of these questions we have some success criteria.

Success criteria

(a) A mechanism which illustrates (data collection and display) how and where projects are different.
(b) A mechanism which links attributes of project launch to project success.
(c) A way of identifying these key characteristics.
(d) A project description notation which shows the key project characteristics.
(e) A mechanism to judge project success (within tight time constraints).
(f1) A plausible hypothesis which relates some aspects of project launch to project success.
(f2) A test of this hypothesis.

2) About the process modelling

   a) Can process modelling using the extended DFD (TRADE) notation provide insights about the launch process which cannot be gained using data flow or other existing techniques?

b) Can process modelling be used to facilitate guiding the launch process?
c) Can the process models be used as a framework for data collection?

Again each of these questions has some related success criteria.

Success criteria

(a1) Some further insights about the nature of the launch phase of the software development process.
(a2) An illustration of how the extended notation shows these aspects of software development, and how they are not apparent using conventional approaches.
(b) An illustration of how the process modelling using TRADE can support identification and education of successful practice.
(c) A way of combining the data and process models such that the models provide a framework for the data collection.

5.2 Instance Case Study Design

5.2.1 Sources of Information

5.2.1.1 For characterizing project launch

The principle way to collect project data, on activities and effort spent on those activities, is in structured interview with project managers. The interview uses a collection sheet which describes launch activities in terms of activity types. This collection sheet will be tried and extended during this study until we have a stable set of activity types. However, the project manager will also be encouraged to give some qualitative description of the project which will be used to accompany our models.

In order to validate the data collected we also examine a number of other sources. These include engineering time-sheets (held centrally), project managers time-sheets, weekly e-mails to and from project managers to engineers to report hours spent, finance records (which record time spent by staff allocated to project codes), project documentation (for example minutes of review meetings, memos about progress etc.), entries in the bug data-base and records of e-mail about the project.

5.2.1.2 For Judging Project Success

Project success will be judged by questionnaire of project managers. In conjunction with staff (engineering project managers, marketing representative and quality manager) at the site we have developed a score sheet of twelve project characteristics considered most important to the organization. Each characteristic is scored from one to ten, one representing extremely poor performance in that category, with a score of ten being regarded as a complete success. The project manager score for each category will be checked against others with knowledge of the project and if the score varies by 3 or more, we will go back and investigate further.

We have investigated a number of sources to validate the success[31] scores against the actual success of the project, but owing to limited resource and access we will only turn to these if we have score disagreements.

Scores of project success if never formally validated are, therefore, entirely subjective. However, we are only using the statistical inference from these scores as a tool to discover a pattern from which we can then pose a plausible theory. It is this theory which we hope to be generalizable to the projects at the site.

5.2.2 Evaluation of Results

Again we intend to present results of the investigation of five projects to the organization. If these projects are found to provide useful insight into the launch process, we expect the organization to move towards collecting this data for new projects. If this happens we will consider the study a success.

5.2.3 Verification and Validation

We will present the models produced to project managers, in order for them to verify that these are a faithful, accurate and representative depiction of their project.

By showing the project time-scale, activity effort and duration, and when the activity occurred, the notation lessens the flexibility of the modeller. In addition, we have already agreed on the level of granularity or detail of the models, and thus there is no choice for the modeller of how to group activities hierarchically. Certainly there is much less room for choice than with standard data flow diagrams. To further validate the models, each will be examined in conjunction with data collected by a colleague external to the organization who is familiar with the notation.

5.2.4 Benefits

As with the initial study we cannot provide quantitative data for the increased understanding or for process improvement. However, if we can provide evidence of further insight into the launch process, for example by identifying key activities and their association with scores of project success, then this potentially benefits all projects at the site. In addition, we can identify problems with projects and differences among projects that are not identifiable using other notations.

5.2.5 Deliverables

We have agreed to the following deliverables:

1) Depiction of the launch processes of the five projects to be studied.

2) Analysis of results to attempt to discover key process activities.

---

31 These include adherence to schedule, extent to which project was over-budget, amount of unplanned overtime, overtime peaks (how much and when and try to correlate with activities), amount of re-work needed later (i.e. more successful needing less re-work) , time charged to project code since it was meant to go to alpha test, and all faults reported against product (since the original product deadline).

3) Mechanism to facilitate the data collection needed to produce the launch process models for new projects. This is to be incorporated within the revised launch process templates to allow minimal impact on the process.

5.2.6 Questions for the Case Study Design

Here we use the DESMET CSDA design questions to consider issues pertinent to the study.

Q1 Baseline against which to compare the results of the evaluation.

A comparison across projects which will be modelled using the extended notation. Again the modelling is post-project, in order to see if we can discover further insights about the development process. Five projects will be modelled, all having already taken place. Although the process modelling is a treatment we have not selected equivalent projects which we will not model. Rather we intend to compare the insights we gain about the five modelled projects, with those from our exploratory study. The exploratory study acts as our control group. In this way our investigation is more focused on our hypothesis; that the TRADE notation does provide further process insights (see Q4).

Q2 Constraints

Our constraints are mostly to do with the limited amount of time allocated to us for access to the site personnel. However, we are also limited by access and availability of data, and by our own time-scales. In brief our constraints are:

Access to information for the modeller.
Availability of data on effort expended on activities in the launch process.
Availability of data relating to project success.

Time-scales for the modeller.
Impact of interview time to organization personnel.

Q3 Hypotheses

We wish to test the following hypotheses:

Process Modelling Hypotheses

H1) Process modelling using the TRADE notation can provide insights about the launch process which cannot be gained using data flow diagrams or without process modelling.

$Treatment_0$:	Baseline. Existing process description.
$Treatment_1$:	Data Flow Diagrams. These treatments form the exploratory study.
$Treatment_2$:	TRADE models. Which is the basis of this later (instance) study.
Response Variables:	Qualitative comparison of the kinds of insights gained using the existing descriptions, data flow and TRADE.
State Variables:	Process modelling using DFD and TRADE allows us to characterize projects in different ways.

Effect Investigating TRADE provides a richer picture than the other approaches.

H2) Process modelling using TRADE can identify key areas of projects that cannot be determined via DFDs or without process modelling.

Treatment$_0$: Baseline. Existing process description.
Treatment$_1$: Data Flow Diagrams.
Treatment$_2$: TRADE models.
Response Variables: Ability to identify key tasks that can be used to predict project success.
State Variables: TRADE: Task effort. Task ordering. Task start and finish (in project days). Average resource usage.
TRADE and DFD: Tasks. Inputs and outputs.
Discussions with project managers and process users.
Effect Investigating TRADE provides a more complete picture of process than the other approaches, and allows us to identify key areas of projects.

Launch Process Hypotheses

H3) Effort spent of key launch activities will be a factor in project success.

Treatments (T_1->T_5): Differing distributions of effort spent on tasks in the launch process of the projects.
Response Variable: Subjective scores of project success.
State Variables: Task effort.
Effect Investigating Impact of effort spent on process launch activities.

H4) The relative amount of effort deployed prior to 'official' project launch has an impact upon project success.

Treatments (T_1->T_5): Differing distributions of effort before and after 'official' project launch.
Response Variable: Subjective scores of project success.
State Variables: Effort before launch. Effort after launch.
Effect Investigating That pre-empting the official launch of the project has an impact upon project success.

Q4 What are the response variables?

The response variables are outlined above (Q3). For our process modelling hypotheses these are qualitative variables. The only quantitative response variables collected are our subjective measures of project success (for hypotheses three and four). (See justification of using a subjective measure in section 5.2.1.2).

High scores = High project success.

(See our comments on analysis in question 15. Our analysis is presented in Chapter Seven).

Q5 What are the experimental objects?

Individual projects are our experimental objects. (see Q1).

Q6 When in the process will the modelling take place?

We will be using process modelling to model the launch phase of the development process. However, our modelling will actually occur post-project for the five projects in this study.

Q7 When are measurements of the response variables taken?

As with the modelling of projects, the collection of response variable data will take place after project delivery.

Q8 What data will be used to allow measures to be calculated?

We will collect effort data, from existing project records, in order to produce models and to have data for our independent variables.

Our response variables will be subjective scores gathered directly from staff.

Q9 Can the effects of the treatment be isolated (confounding factors)?

Confounding Factors

Staff

Staff were allocated to projects as normal in the organization, and our case study had no control over this. All of the project managers are those who are happy to be involved, i.e. those who see the worth of the study. As we are comparing across this selection, rather than against projects outside this selection this will not bias our results.

It was expected that the project manager would be the biggest confounding process factor, though in fact two very different projects (one success and one not very successful) were managed by the same person.

Time

The modeller was using the method over the same period of time. Therefore, there will be no learning effect on site staff to bias the results.

Project Factors: Difficulty or Complexity, Schedule

There was a deliberate choice of both large complex projects and smaller projects, and also of projects with tight or not so tight schedules. This would enable the investigation of alternative hypotheses such as the size, complexity or schedule being the main impact on project success.

Initial Investigation. The only alternative hypothesis is that in getting a group of people to discuss their process in a structured manner they have understood it better. But then this is part of the point of process modelling anyway, and is not really an alternative.

Q10 Procedures to ensure method / tool used correctly.

This method is essentially the invention of the modeller, hence, the main procedural task is to ensure that it is used consistently. In order to do this

each model has been examined by the quality manager, and by another colleague with knowledge of the study and the new notation.

Q11 To what extent will the treatment be integrated into the current process?

It is our intention to integrate the data collection into the launch process, after this study. However, inclusion in the process template may increase effort. Therefore, our collection sheet is designed to have the minimum impact, and to record the impact (time-taken) that the activity took. All measures will be calculated automatically.

Q12 State variables / project characteristics which are important to the study.

This work examines five projects. Projects were chosen (by the organization) as being typical, with the stipulation that they did not just pick, for example, successful projects. Their characteristics are:

Application area:	All the same. Software for inclusion in board test equipment. Must be applicable to these.
Methods / tools:	There is no imposed development method (e.g. Yourdon) at the site.
Development process:	All projects should follow the same (launch) process under investigation.
Scale of project:	One much larger project, three of same scale, with one software release (again similar scale to the three).
Project managers:	Quality and experience of Project Manager is potentially a large influence on success. For our five projects we have four project managers, all of whom are experienced, and highly regarded within the organization.
Project Complexity:	One project (Z) was more complex, large and important than the other four. One project was less important[32] (W). The other three projects were of a similar nature.
Own Characteristics:	Our own characteristics of project launch, are depicted by the TRADE models, from which we hope to be able to make some useful project comparisons. (See also our description of hypotheses in Q3).

The state variables for each hypothesis are given in our description of those hypotheses in question three.

Q13 To what extent do we need to be able to generalize the results?

Results need to be applicable to projects at the study site.

---

32 Note that the TRADE model of project W (see chapter six) reflects its relative lack of importance, showing a gap of several weeks where no work on this project took place, due to resources being switched to other projects of higher priority.

There is no baseline, other than the exploratory models. We can only informally compare projects with other projects modelled. We will use numbers, e.g. effort measures and success factors in order to help to validate qualitative opinions and to discover interesting patterns among the project data.

For project success factors (response variable) we will use a median score of all factors, and compare each project's individual factors with a cross-project median.

Q14 Confidence in evaluation results

We have tried to choose five projects which are representative of projects at the site (see section 7.2 which considers the extent to which this has been possible). We believe that the patterns and results that we see for these five projects will be representative of the majority of projects at this site[33]. However, again, this study is intended primarily to show the worth of a particular approach. If successful we intend to collect data across all projects at the site (see Q11).

Q15 How to analyze the results of the evaluation

For project success scores (response variable) we will use a median score of all factors, and compare each project's individual factors with a cross-project median.

We intend to use non-parametric tests (e.g. Spearman Correlation) to investigate the associations among success scores and effort spent on characteristic project launch activities.

Q16 Appropriate level of confidence?

We expect that the projects which have been chosen are representative of projects at the study site and that the results from them will also be representative of projects at the site (see Q14, and section 7.2 on the limitations of our work).

5.3 Instance Case Study Conduct

5.3.1 Modelling Strategy or Framework

We had already produced models of the theoretical process in the previous case. The instance models were to be based solely on interview and observation. Hence, the production of models depicting actual projects involved no prior study of documentation (of which the modeller was already aware); simply interview and a further (shorter) validation interview.

A related task for the modelling was to refine the data collection framework which was based upon activity types. Hence, all interviews used an initial data collection form which was revised and updated before the next interview. This form became

---

33 Our discussion in section 7.2 clarifies this claim by suggesting that our results are representative only of non-specials projects with project managers who are not hostile to our modelling technology.

stable after three interviews and it was then clear that we had a stable set of activity types.

Therefore, our framework for modelling (see below) was actually much simpler than for the exploratory case.

1) Discern activity types for data collection sheet (Used previous process knowledge).
2) Arrange interview(s) with project manager.
3) Produce sheet for data collection (interview).
4) Conduct interview about project.
5) Produce project model.
6) Revise collection sheet.
7) Validate model and changes to collection sheet by further interview.
8) Ascertain success scores for project (may involve other staff).
9) Repeat from 2 for next project.

5.3.2 Modelling Issues, Thoughts and Issue Resolutions

5.3.2.1 Process Models as a Data Collection Framework

It has been suggested [17, 18, 51-53, 154, 170, 181, 243] that one of the advantages of process models is that they can be used to show what data to collect and where in the process this data is to be collected. However, our study wished to examine both differences in what activities took place in a project launch, and in what effort was used by these activities. This presented a dilemma. If we assumed a particular process model and collected information against its activities, then we would unduly influence the picture of process which we wanted to be told about, in that we would not learn about differences in project activities. Conversely, if we did not assume some process framework, then we had a problem in deciding what we were collecting about or against. (For a 'one-off' study we could produce a model, and then collect information against it, but this would never allow us to move towards data collection being automated, or towards staff at the organization collecting information without further modelling input).

We found that although projects varied greatly in the way they were carried out, the mandatory production of types of documents using associated templates (see both Chapter Four and Chapter Five for a description) effectively meant that we had projects which had a large number of activities, which could be classified as belonging to a relatively small set of activity types. For example, a commonly used document template was the product proposal (see Chapter Four). This document could pass through any number of lettered and numbered revisions[34], some projects might have only one revision, whereas others might have ten. Once this idea of activity types had been established, it became clear that it could be used as a collection mechanism, because, for example, collecting against a product proposal revision type, would not stop people from telling us about the number of revisions, their real sequence (e.g. A, C, B), when they were produced, when they were agreed, and so on. In addition, such terminology was consistent with existing procedures which meant that we could again fit in with organizational culture, and that most importantly, staff encountering the data collection terminology would already be familiar with it.

---

[34] The distinction between these types of revisions is centred around the formality of the documents review process, however, understanding of the precise review mechanisms is not necessary to see that the document can have 1 to n revisions.

5.3.2.1 Data Collection

Having a template for collecting data about project activities implied a process. In effect, the activity types against which we collected information would be a generic process model. The organization were keen to start collecting measures for real projects as soon as possible, However, it was clear that the format for data collection must be right before we started collection on new or existing projects. Thus, we opted for a trial approach, in that we wanted to refine our collection strategy by examining past projects before changing the process (and templates) to collect information on new software projects. It was important to have a complete and stable set of activity types which was as accurate and representative as possible, and encompassed the range of projects at the site. Therefore, the choice of which projects to investigate was potentially very important. Classic experimental design would necessitate our attempting to eliminate all variables other than changes to the activities used by the process, in order to reach some conclusions. However, this was not practical for our study. Similarly, we might wish to look at two similar sized projects, with the same project team, under similar time-scales. Once again, we had a contradictory goal in that we also knew that it was important to look at a wide range of projects. To satisfy this goal we would ideally choose four very different projects:

	Large	Small
Successful	Proj 1	Proj 2
Problematic	Proj 3	Proj 4

Again we were searching for a compromise. In discussion with the organization, it was felt that the biggest (controllable) influence on process, and possibly also on project success, might be the project manager. It was, therefore, decided to look at some projects with different project managers (different process), but to also try to look at two or more projects with the same project manager. Thus, of the four remaining projects which we investigated two (one successful and one not successful) were managed by the same person[35].

5.3.3 Mechanism for Using the Models

There are two ways in which the project models are compared. The first is by qualitative inspection. For example, we examine the 'shape' - determined by the ratio of the height (resource usage) of the activity rectangle against its length (duration) - and size of equivalent activities across projects. (See a more detailed discussion of our notation in section 6.1). We examine the overall picture of a project. For example, are there lots of stretched out activities - indicating low average effort usage with proportionally long durations, or more tall thin activities - indicating shorter durations with proportionally higher average resource usage. We look at which activities occurred before the official project launch, and we look at how many iterations there are of each activity type.

The second way we compare the models is to examine the data which is used to form them. Some of this can be seen just as easily on the model itself, for example, duration of activity (working days), average resource usage for the activity (in man

---

[35]This is by no means, however, the full story, in that many other factors, size of project, availability of project managers to be interviewed, their willingness to co-operate, the likelihood of data being available, or even of them remembering about the project all had an influence.

days), total effort for the activity (man days). However, we also calculate a number of other measures, for example, the percentage of launch effort spent on the initial version of the product proposal and the percentage of effort spent prior to official project launch (see below). Finally, we compare elements of the models with the success factors for that project, in order to try to discover whether key activities have any effect on project success (see following chapters).

Measures	Projects					
	V	W	X	Y	Z	Units
% effort on product proposal A	7.4	18.2	25.2	0.2	30.1	percent
% effort on project plan	1.9	20.5	8.4	0.2	6.0	percent
Raw launch activity effort before day 0	3.5	5.3	10.7	28.0	22.7	person days
Total launch activity effort	27.1	22.0	44.7	33.6	108.0	person days
% of launch activities effort before day 0	12.9	24.1	23.9	83.3	21.0	percent

5.4 Evaluation of the Instance Case Study

5.4.1 Successes

The success of the study was that it did provide further kinds of insights into the launch process which we could not have previously found, owing to the limitations of existing notations. We believe that we validated the hypothesis (H3) that "effort spent of key launch activities will be a factor in project success'. In addition, we gained further insights, specifically concerning the nature and extent of differences among projects, which we had not envisaged. (We describe these further insights in the following chapters).

5.4.2 Failures

5.4.2.1 Comparison of Reference and Instance Models

As with the previous case we had hoped to provide a further two streams of modelling, one depicting the theoretical process and one the depiction of real projects. These two streams of modelling (named reference and instance models) were to be entirely separate. The reference model would be a description solely based upon documentary descriptions of process. This had three purposes: to suggest improvements to existing process descriptions, to act as a basis for a new reference model and to be used as a basis for comparison with instance models. The reference model exercise did not involve any collection of effort or time data, and was in essence a repeat of the first phases of the exploratory case again using standard data flow diagrams. The exercise took only approximately two to three man days effort in total, it was completely separate from the examination of projects, and really forms a minor extension to the exploratory case. We did discover some reasons for why process perception was a problem during the reference model exercise, but there was nothing that could not have been ascertained from our initial case study. Furthermore, the difference in the level of granularity between the reference model and the project (instance) models was such that we were unable to make any meaningful comparisons between the theoretical and actual processes.

5.4.2.2 Data Collection

We started this study by conducting an investigation to find out what project data was available that we could use to help quantify our process models. This investigation managed to find very little other than that it was going to be very

difficult to collect any data on projects, including effort data. This blind alley was terminated when we realized that we needed to return to first principles, i.e. to define goals, then questions, then measures, and to decide how we would use these measures. When we did this and approached people about collecting the same effort data, they did manage to find that such data was available. Furthermore, convinced that we were going to use the data in way that might be of some benefit to them, they made an effort to ensure that the data was as accurate as possible.

5.4.2.3 Data Validation

Ideally we would wish to have some simple measures which could be used to validate the subjective project success scores, for all of the projects studied. Although budgets and time-scales were checked for inconsistency (e.g. we could check records of a project's progress) we were not able to do this for all project characteristics. A particular problem example is ascertaining customer satisfaction. Although we could, and did, question marketing representatives to validate project manager scores we were not able to talk to actual customers. Finally, we abandoned such attempts at validation, convinced that the striking agreement on project scores added sufficient weight to their validity.

5.4.3 Cost benefits for future usage?

As with the initial case study we can only conjecture that the identification of key process activities will lead to a reduction in cost. This study has had some blind alleys and inefficiency, but has still cost the organization only ten hours of staff time plus approximately forty hours of our time. In addition, (for these five cases) we have occasionally had to look quite hard to find the required information. However, having completed this study, data collection procedures will form part of the templates for production of launch process documentation, and all measures automatically calculated. This means that the impact of collecting this information will be minimal, perhaps 20 or 30 minutes per project. The project managers involved in the study all remarked that now that they know what they will be expected to collect that it will be very easy. Despite this minimal effort, the potential benefits are significant, in that reductions to project cost are likely to of the order of days or weeks rather than hours.

As we have remarked in Chapter Four (section 4.4.3) the early detection of problems is extremely cost efficient, because the cost of fixing problems late in the process may be as much as 100 times greater than if they had been detected early (e.g. in the launch phase). Our own figures suggest that for one project examined early detection of a design problem would have saved over 60 person days. Even if we only detect one major error in every five projects, we would be spending perhaps 2 to 3 hours in order to save at least 480, a saving of 160 times. Even assuming that increased understanding does not produce quite such a high rate of detection, the potential cost benefits are still significant.

5.4.4 Recommendations for future use

5.4.4.1 Critique

This later study of projects evolved from our earlier work, and from the realization that we needed to adapt the modelling techniques in order to gain further insights into the launch process. In addition, the instance study was much more geared to satisfying organizational needs and to providing the site with information about their projects than the earlier exploratory study. The success and the increased visibility after our presentations meant that there was a heightened expectation

within the site, and much enthusiasm for the work. As a consequence the schedules for work were much tighter, and there was much more emphasis on getting information rapidly.

Hence, the data gathering for this study had less time per project than for the exploratory study. Similarly the drive to produce a collection system which could at some later point be automated, meant that collection during this study was moving towards a more automated approach. This meant that there was less time spent by the modeller discussing and validating measures, and that there was greater emphasis on being provided data by the project managers. If carrying out such work again we would wish to have stronger data validation procedures, and greater time allocated for investigation of corroborating evidence (e.g. time-sheets, e-mail, project records).

Despite this caveat we still managed to discover some useful patterns in our models and our data, which suggested where launch effort is most essential. Furthermore, the most striking (though to us surprising) findings of this study are not dependent on the actual data. We started the study to find out something about key activities, and devised a notation to do this, but actually found out something else, which we believe to be more interesting and more important to the software engineering community. Thus, our study was a success, but not really in the way that we had intended.

5.4.4.2 Lessons Learned

As with the initial (exploratory) case we have compiled a list of additional lessons learned from our experiences of modelling during this study.

1) Try to avoid the time-period between interviews being compressed through problems with staff availability or time-pressure.

   For the exploratory case study, one of the most successful strategies was the approach taken to interviews. We had allowed a day for each person, so that in each case a model could be constructed, or a previous model revised, and questions could be more efficiently targetted for the next interviewee. For example, if part of the process or project was still not clear then this could subsequently be given more emphasis. However, for the first process instance, limited staff availability (due to travel arrangements, visits to other sites etc.) meant that five people were interviewed in less than two days. This did allow sufficient time for remodelling and re-emphasizing of questions for the next interview. Consequently these (first instance) interviews were less efficient (and less revealing) than those of the exploratory study (and yet in each strategy the time with each individual, which was the main impact to the organization, was the same).

2) Be consistently available or on-site so that people know how and where to contact you.

   Being on-site, as often as possible, and ideally being on informal terms with people at that site, allows you to absorb much more of the organizational culture. However, pressure of time may not always allow this. As a minimum, it is good to have some time, say once a week (or fortnight or whatever), when people know that you'll be working at desk X. Often someone has thought of something which 'might be relevant', but which perhaps (from their point of view) is not worth contacting you about. These 'chats' can be very valuable, for example in suggesting sources of information, and in eliciting more genuine responses to your proposals or ideas.

5.5 Use of the Instance Case Study

This chapter has suggested that the instance case study has shown that the extended TRADE notation does provide unique insights into the launch phase of the development process. We believe that we were able to determine the activities which have greatest impact on project success, and were able to uncover other interesting findings about the projects at this site. However, rather than go into greater depth here we will postpone the analysis of our findings for a further chapter. Hence, Chapter Six gives a fuller description of the models produced and the data used by those models. Thus, we illustrate our collection mechanisms and our modelling notation by reference to examples taken from this study. We then present those models and the results of our process measures. Our analysis of the results and discussion of our findings follows separately in Chapter Seven.

6. Description of Results

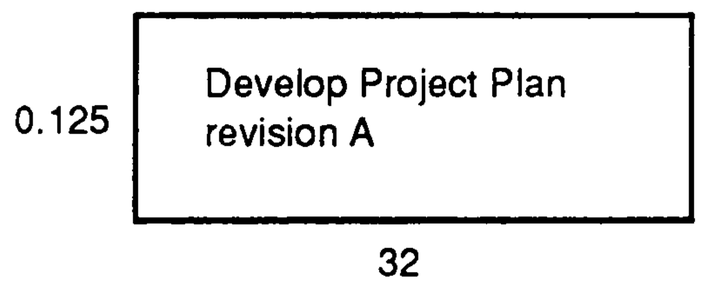
Chapter Synopsis

This chapter describes the models and data produced during our studies, and some of the results from our modelling work.

6.1 Models

The modelling notation used by the exploratory study was data flow diagrams. All of the models produced by the exploratory study can be found in Appendix A. However, we used an extended notation (TRADE) for the later studies. Since this notation (though briefly outlined in Chapter One) will be unfamiliar to the reader we give a brief description of the elements of the notation, before showing the models which we have produced using it.

The first change is that we use a rectangle to represent the activity rather than an ellipse or circle. This is so that we can scale the rectangle along two axes. The horizontal (x) axis is the duration of the activity and the vertical axis (y) is the average effort during that duration. A simple example of the activity representation is given below.

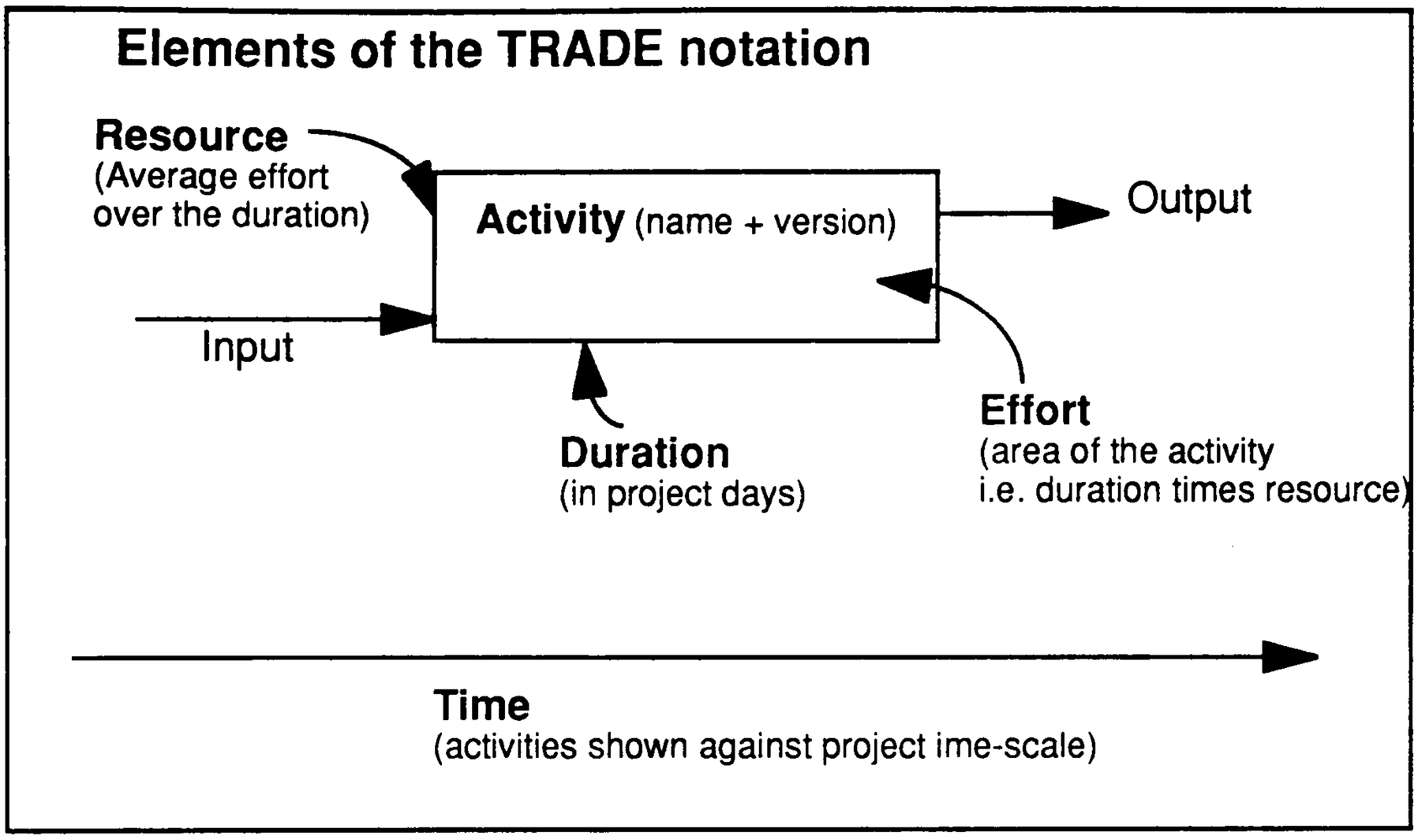


The activity can be seen to have lasted x days (horizontal axis). The vertical axis (y) shows the average resource usage for the activity. However, the main advantage of this scaling is that it allows the area of each activity (x times y) to faithfully represent the total effort (in man days) for each project activity. Therefore, this activity took place over a duration of 32 project days. The total effort expended was 32 * 0.125 = 4 person days during the 32 day period. The advantage of having both horizontal and vertical scaling is that it gives us a further insight into the pace or urgency of each activity. For, example we might have two other activities with the same total effort used but where the distribution of effort was very different (see below):

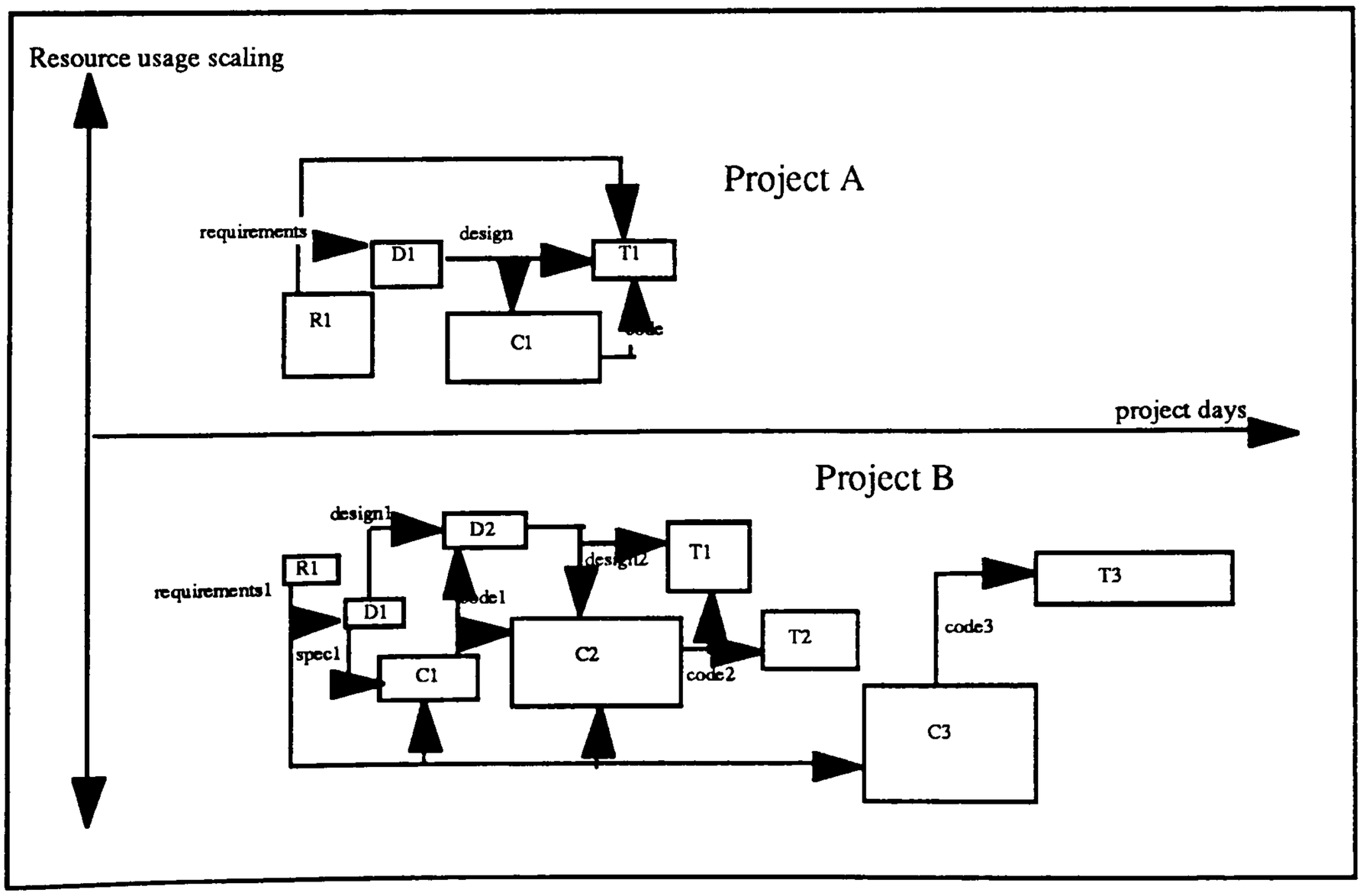
	Duration	Average Resource Usage	Total Effort
Activity A	32	0.125	4
Activity B	8	0.5	4
Activity C	1	4	4

Each activity is placed on a time-scale which shows when it took place during the project (in project days - see later). As with standard data flow, activities are also connected by the inputs and outputs among them. When we have a number of activities which formed the project launch these patterns of resource usage can be seen on a larger scale. For example, lots of tall activities (high average usage and short duration) with short gaps between them, would imply that the project had a

different priority (or schedule to match), than long thin (lower average effort for each day) activities. Similarly, we can see the different emphasis given to different activities within a project. The advantage of the models is that this kind of information can easily be seen by humans, without the need to consult the underlying figures, and they can easily see patterns and form impressions of project launches.



We will describe some of the other features of these models in examining the results of our modelling efforts in coming sections. However, we first present a simple example. This example assumes a simplified version of the software development process. Our highly simplified process consists of the following activity types; production of requirements and specification documents (R), design (D), coding (C) and testing (T). Each version or iteration of these activity types is numbered (e.g. the first round of requirements and specification activities is denoted R1).



We illustrate above two projects, modelled in terms of these activity types. Note how the first project (A) has proportionally more effort devoted to the first iteration of requirements and design activities, but far less effort overall. The second project is typical of a more chaotic process, where effort is ramped up near the end. It has a number of coding and testing phases with increasing effort devoted to each. Though this a simplified example it is based on our experiences of projects at the study site. Indeed, our experience suggests that actual projects may be far more different than this simple example suggests. Note how the use of our TRADE notation shows the differences in the total resource usage for each project, the relative amount of resource used by activities, and the extent to which certain types of activities are revisited (iteration).

6.2 Results

We have used a number of sources of information in this study, for example, procedures documents, interviews, artifacts. However, owing to the confidential nature of much of this information the evidence which we shall present comes primarily from our own notes, reports and models. In keeping with the rest of this thesis we have opted to consider the evidence from the exploratory study and the instance study separately. However, we concentrate mainly on the latter study, which employed a unique modelling technique to discover insights into the launch process.

6.2.1. Exploratory Study Evidence: Questions and data relating to those questions

The exploratory study evidence came from the following sources.

1) Notes and preliminary models which were produced during the modelling of the theoretical process.
2) Notes and models produced from the interview phases of the exploratory study.
3) Notes made on the interviews themselves.
4) The report on the study which was presented to the collaborating organization.
5) The presentation of our findings to the collaborating organization.
6) Anecdotal evidence, including comments made to us and our observations.

Unfortunately much of our findings are quite explicit and confidential. Therefore, we cannot give details of interviews or of the report which we presented to the organization. However, models produced during the exploratory study can be found in Appendix A.

We have already shown (Chapter Four and Chapter Five) how the planning stages of our studies were linked to a number of goals and questions. In considering the results of our work we thus try to link these results to the questions which we had initially posed. For our exploratory study we had posed the following questions. (Note we shall consider whether we have satisfied or refuted our hypotheses in the following chapter on analysis and findings).

1) About the process

   a) Is launch process a suitable candidate for study?
   b) Do the users really know what the standard process is?

Is deviation by ignorance or by design?
Does it matter?

2) About the process modelling

a) Can process modelling help us to understand the process?
b) Is the notation used (DFDs) a viable choice?
c) Is the strategy adopted a viable choice?

6.2.1.1 Answering questions (Models and Interviews)

Answering all of the above questions (and providing their related success factors - see Chapter Four) hinges on the production of process models which can be used to identify and explain discrepancies between the theoretical and actual processes. As we have described (Chapter Four) the analysis of our models was based upon standard heuristics for the examination of data flow diagrams, and upon subjective analysis. For example, where we produced a model which had an unconnected activity or flow we would then re-examine procedures or re-configure our questions in order to find the reason for this. The new data flow model could then be used to make the intended connection much more obvious and clear to process users.

However, it was the activity of modelling itself which (by forcing the modeller to gain a more complete understanding of process - through studying procedures or interviewing process users) revealed insights into the reasons for process deviation. For example, one of the key findings from our interviews was that users felt that the existing process was too detailed, and thus often decided to skip certain sections of documents. It became clear from further interviews that this was a problem of perception or lack of understanding. The process designers had intended the process to be quite iterative, so that successive versions of process documents would include increasing detail as the project progressed. When we re-examined procedures we found that the only (and at that rather cryptic) mention of the iterative nature of the process was over half way through the procedure, where the document produced by the phase was referred to as: *'a " living document " that is expanded in defined steps to meet the specific and defined requirements of the Management Review Team'*.

6.2.1.2 Results of the Study

The termination of the exploratory phase of our work was a presentation to senior management and to process users involved in the study, in which a number of our key findings were presented. These included that:

1) There were many[36] discrepancies between the documented and actual processes.

   It was clear that the view of the process which its designers had, and the view of the process its users had were very different (see the example below about iteration).

   The benefit of the exploratory study was not only that the modelling discovered and highlighted discrepancies but also that the models were an excellent medium to illustrate these discrepancies.

---

36 We presented six key findings at the pilot presentation, and noted around twenty specific problems in our report.

2) A major part of the process (the entire concept phase) was found to be ignored[37].

3) The phased or iterative nature of the launch process (which was suggested by the procedures documents) was not clear to process users.

   For example, process designers expected that there could be a number of versions of same document with increasing levels of detail, whereas users took a much more rigid view of what they were expected to do, and hence they often questioned the amount of detail required.

4) The process was not being used as intended. It should have focused on business decisions but actually served design much better.

It is interesting to note that many of the problems were with process perception, and not process design - see the previous section. This suggests that one of the uses for process models is in providing process guidance (e.g. adding clarity to procedures).

Other benefits of the exploratory work included:

1) The process of modelling helped to identify some specific problems with procedures. For example:

   a) Activities clear but not their sequence.

   b) Interactions between activities unclear.

   Interestingly these (a and b) are features which the data flow notation does not illustrate well, and yet the discipline of modelling forced the modeller to consider which activities preceded others, what outputs from one activity were necessary for another and so on.

   c) Recipients of documents unclear.

   d) Contradictory descriptions of the same activity or group of activities.

   e) Different terminologies used for the same document.

   f) Connections between different procedures documents, and between procedures and templates were not at all clear.

   However, these (d-f) are features which the data flow diagram makes very clear. For example, a flow which goes to no activity (which thus raises the question about who, or what activity, is the recipient of a particular document) is very apparent on a data flow model.

2) Identification of specific problems with document templates.

   This was mainly the noting of ambiguities in the hidden text guidance given.

3) Identification of other general process problems. For example:

---

[37] In fact there was only one example of this part of the process having ever been used.

a) Users felt that the process was too detailed, particularly in the early stages.

b) Users wanted to use the process to get projects going, not to support business decisions.

c) Users tended to rely on templates for guidance, and did not refer to procedures. In some ways, by providing extensive hidden text guidance they had proved to be too useful, and now were relied upon almost exclusively.

6.2.2. Instance Study Evidence: Questions and data relating to those questions

The instance study evidence we shall refer to comes from the following sources. (Models produced by the study can be found in Appendix B).

1) Notes and preliminary models which were produced during the modelling of the instances. For the first project examined there was a great deal of information collected which was not used by this study, for example, notes on the availability of effort and success data.
2) Effort data collected for projects.
3) The models produced by the study.
4) Our project descriptions (based on interview) which accompany the instance models
5) Anecdotal evidence.
6) Success scores from project managers.

For our instance study we had posed the following questions.

1) About the process

   a) How are some projects different from each other?
   b) Does process affect project success? If so what is it (in process terms) that some projects do that makes them more or less successful than others?
   c) What are the key project launch characteristics?
   d) How do we describe these aspects of projects?
   e) How do we judge the success of projects?
   f) How can we relate the success of projects to launch characteristics?

2) About the process modelling

   a) Can process modelling using the extended DFD (TRADE) notation provide insights about the launch process which cannot be gained using data flow or other existing techniques?
   b) Can process modelling be used to facilitate guiding the launch process?
   c) Can the process models be used as a framework for data collection?

We now describe the models of the projects and how they illustrate differences among projects. We show how we collected effort data on launch process activities in order to investigate the effect of effort spent on those activities on project success. Finally we describe our mechanism for judging project success. The following chapter will give some analysis of this information, and will try to provide further answers to those questions posed above about process modelling.

6.2.2.1 Data collected

There are two types of data collected. The first is the effort data on projects, which we use to produce our TRADE models. This is collected by interview with project managers, utilizing a collection sheet which shows activity types (and version numbers for those types), start and finish dates for each activity, durations, and effort. All other measures are automatically calculated. We include below an example of the kind of information which we collect and show.

Activity Name	Finish	Duration	Effort	Usage
Develop pdp rev A 5/8/92	17/9/92	31	11.25	0.36
Develop pdp rev B 17/9/92	2/10/92	11	4.25	0.39
Develop pdp rev C 2/10/92	9/10/92	5	3.19	0.64
Phase One Review 4/12/92	4/12/92	1	2.00	2.00
Follow on reviews 20/1/93	20/1/93	1	1.50	1.50
Develop pdp rev 1 20/1/93	8/2/93	13	1.50	0.12
Develop PSO plan 17/9/93	18/12/93	66	0.50	0.01
Develop pjp rev A 5/8/92	18/9/92	32	3.75	0.12
Develop pjp rev 1 8/1/93	27/1/93	13	0.50	0.04

The second kind of data collected is the scores of project success (see Chapter Five for our rationale for adopting this subjective approach). These scores were collected from the project managers responsible for the projects. We will further examine the scores for each project in our analysis and findings. However, we show below the raw scores for each project.

Success Scores for Projects V - Z

Project V											
		Bad				Average					Good
Characteristic		1	2	3	4	5	6	7	8	9	10
Budget						1					
Customer satisfaction								1			
Forced changes to design			1								
Keeping to specification					1						
Management of Risks							1				
Panics									1		
Post-integration bugs				1							
Re-work				1							
Requirements problems						1					
Schedule						1					
Unexpected problems				1							
Unplanned over-time				1							

Project W											
		Bad				Average					Good
Characteristic		1	2	3	4	5	6	7	8	9	10
Budget									1		
Customer satisfaction							1				
Forced changes to design						1					
Keeping to specification											1
Management of Risks						1					
Panics								1			
Post-integration bugs										1	
Re-work								1			
Requirements problems											1
Schedule			1								
Unexpected problems								1			
Unplanned over-time											1

Project X											
		Bad				Average					Good
Characteristic		1	2	3	4	5	6	7	8	9	10
Budget									1		
Customer satisfaction								1			
Forced changes to design								1			
Keeping to specification									1		
Management of Risks							1				
Panics								1			
Post-integration bugs								1			
Re-work								1			
Requirements problems										1	
Schedule								1			
Unexpected problems								1			
Unplanned over-time						1					

Project Y											
		Bad				Average					Good
Characteristic		1	2	3	4	5	6	7	8	9	10
Budget		1									
Customer satisfaction			1								
Forced changes to design			1								
Keeping to specification										1	
Management of Risks				1							
Panics								1			
Post-integration bugs		1									
Re-work		1									
Requirements problems										1	
Schedule								1			
Unexpected problems			1								
Unplanned over-time						1					

Project Z											
		Bad				Average					Good
Characteristic		1	2	3	4	5	6	7	8	9	10
	Works										1*
Budget	Development								1*		
Customer satisfaction									1*		
Forced changes to design						*	1				
Keeping to specification										*	1
Management of Risks							*	1			
Panics						1			*		
Post-integration bugs								*		1	
Re-work							*		1		
Requirements problems						1		*			
Schedule									1*	*	
Unexpected problems								1			
Unplanned over-time				1*							

6.2.2.2 Projects

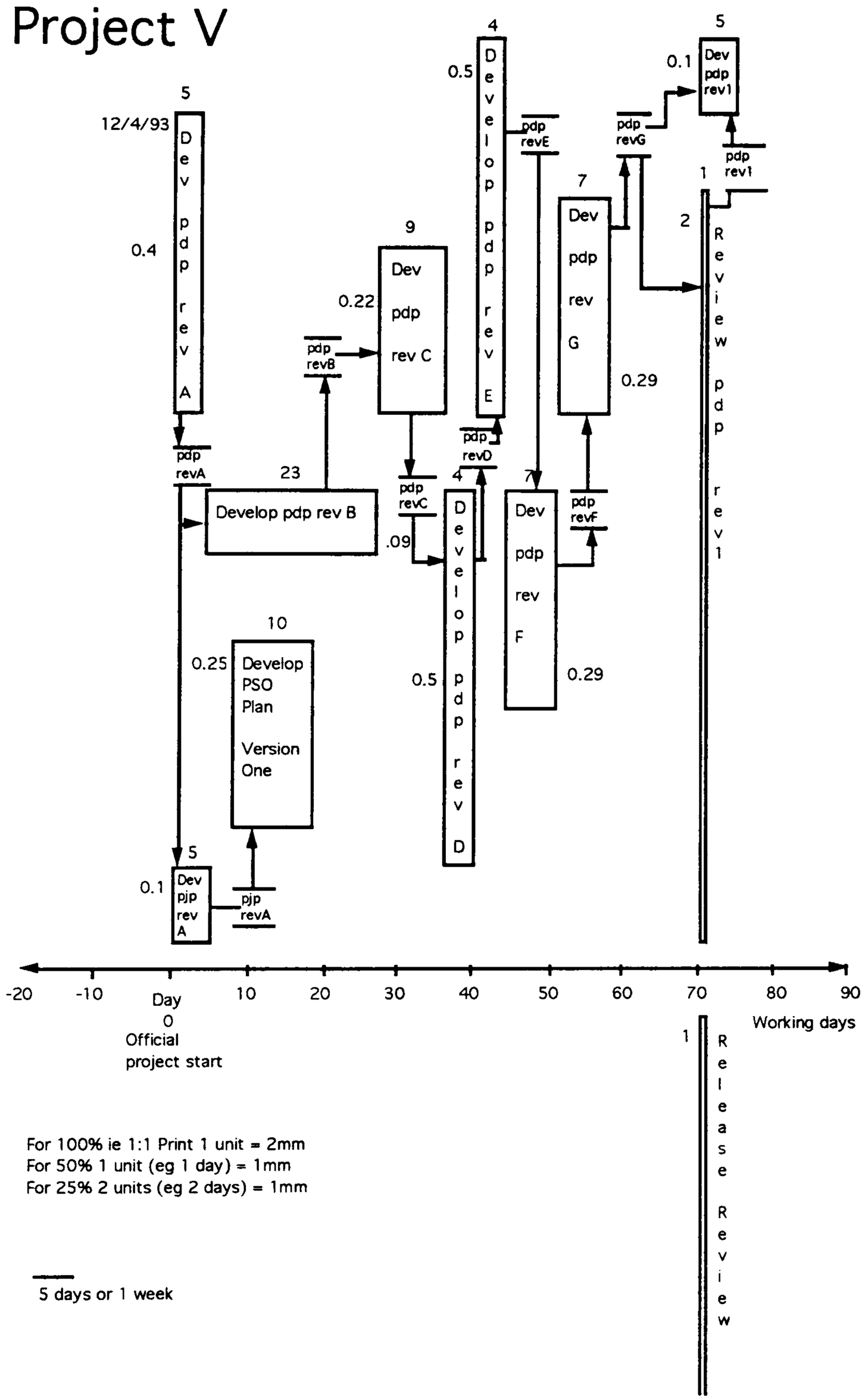
We now briefly describe each project and show the associated TRADE model. Note that for some models we show only a portion of the models and others we show at a reduced scale. The full models can be found in Appendix B.

Project V

Project V was unlike many other projects in that it was a software release. The purpose of this project was to co-ordinate software (from other projects within the organization) into a periodic product release. Thus, even the production of the product proposal document is really only a 'cut and paste' of information from the various contributing projects. The picture of the project (instance model) reveals the nature of this project. The launch consists almost entirely of 7 versions of the product proposal document, which can be seen as pdp revisions A to G. Unusually, these all took a similar effort to produce (being simply a collating of information as the decisions as to what to include in the software release changed during the project life-time). The success of the project was mixed, with a substantial amount of rework resulting from post integration bugs.

Project V Instance Model: 50% Scale

Horizontal : Each unit = 1 day (actual time axis).
Vertical : Each unit = 0.01 people per day (only a scale).



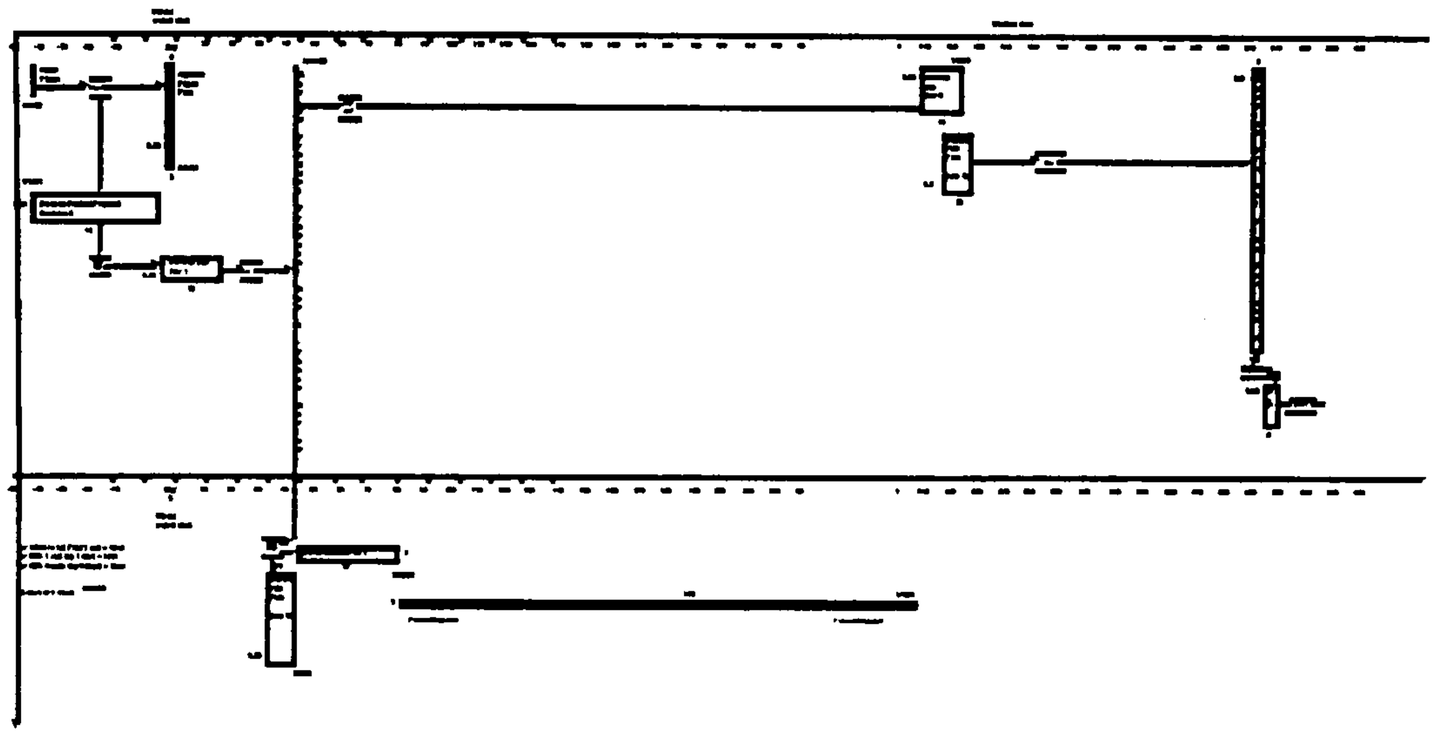
For 100% ie 1:1 Print 1 unit = 2mm
For 50% 1 unit (eg 1 day) = 1mm
For 25% 2 units (eg 2 days) = 1mm

5 days or 1 week

Project W

This was a relatively low priority software project, which involved an upgrade to an existing product. Thus, the project lasted a long time not because it was large or complex but because it was 'put on hold' when resources were transferred to other higher priority projects. This lack of priority may account for why schedule was a problem though all other success factors were high. Indeed, the instance model reveals this aspect of the project, with a number of horizontally elongated activities, each with low average resource usage and spread over a long period of time.

Project W Instance Model: 12.5 % Scale



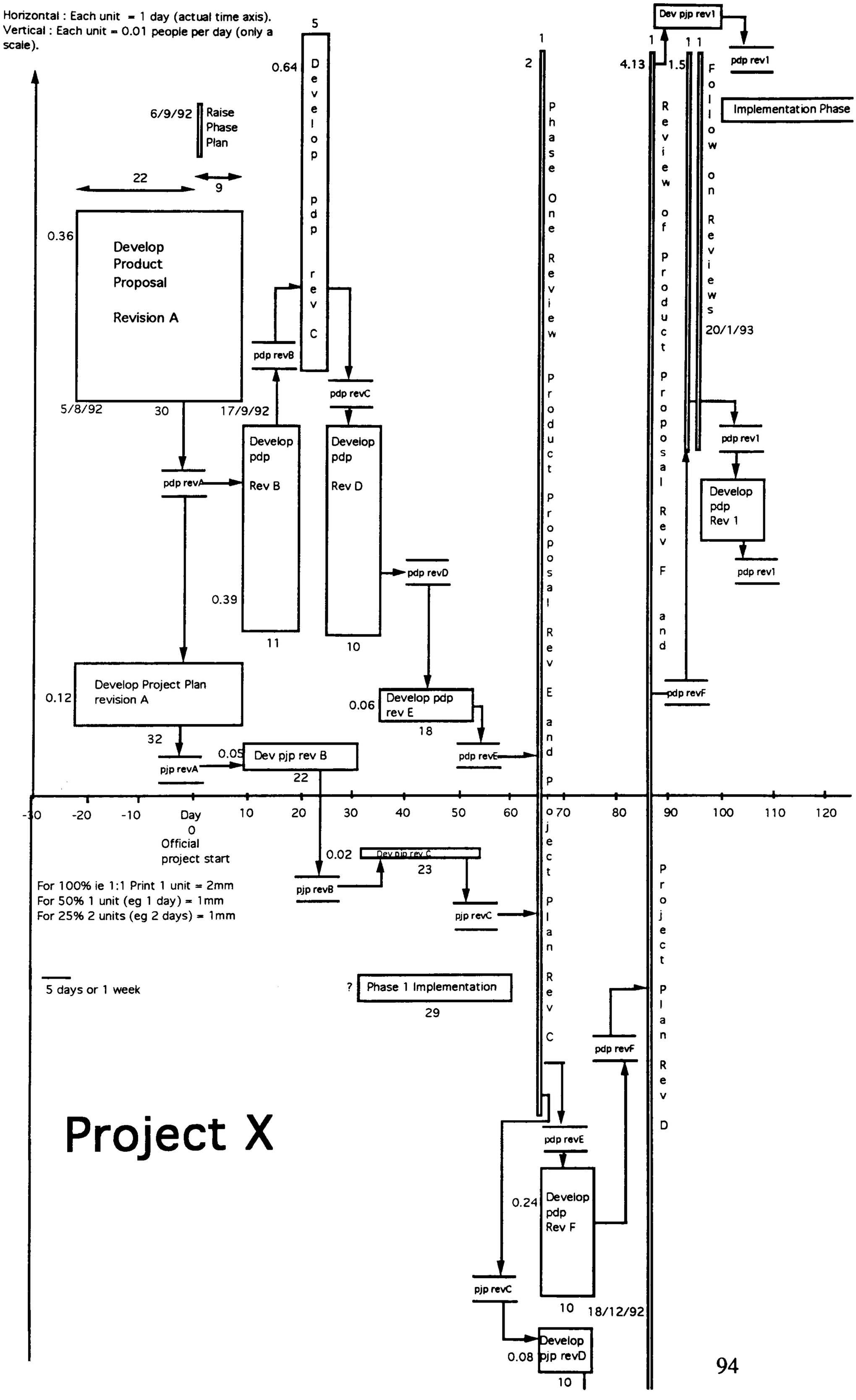
Project X

This was a software project which was to be used by part of a larger project (Project Z). The instance model (or project profile picture) gives the impression of an ordered project launch, with a number of versions of the product proposal and project plans, with the first version of both being the more significant effort (the product proposal revision A being 25% of the total launch effort). Only work on the initial versions of the project plan and product proposal took place before the official project launch (day 0). The activities of the project launch are quite tightly bunched together, and are often tall and thin implying a high resource usage, and a tighter schedule. There is high degree of both parallelism and cohesion between the versions of the product proposal and the project plan.

The project was seen as highly successful; the most successful of all five projects examined; with very few problems encountered (unplanned overtime being the only score of less than six on the success factors). The experience of this project tends to suggest that spending time on the initial product proposal, and not rushing into implementation too soon, is the way to proceed.

(The following instance model diagram shows the key aspects of the Project X Instance Model at 50 % Scale)

Project X

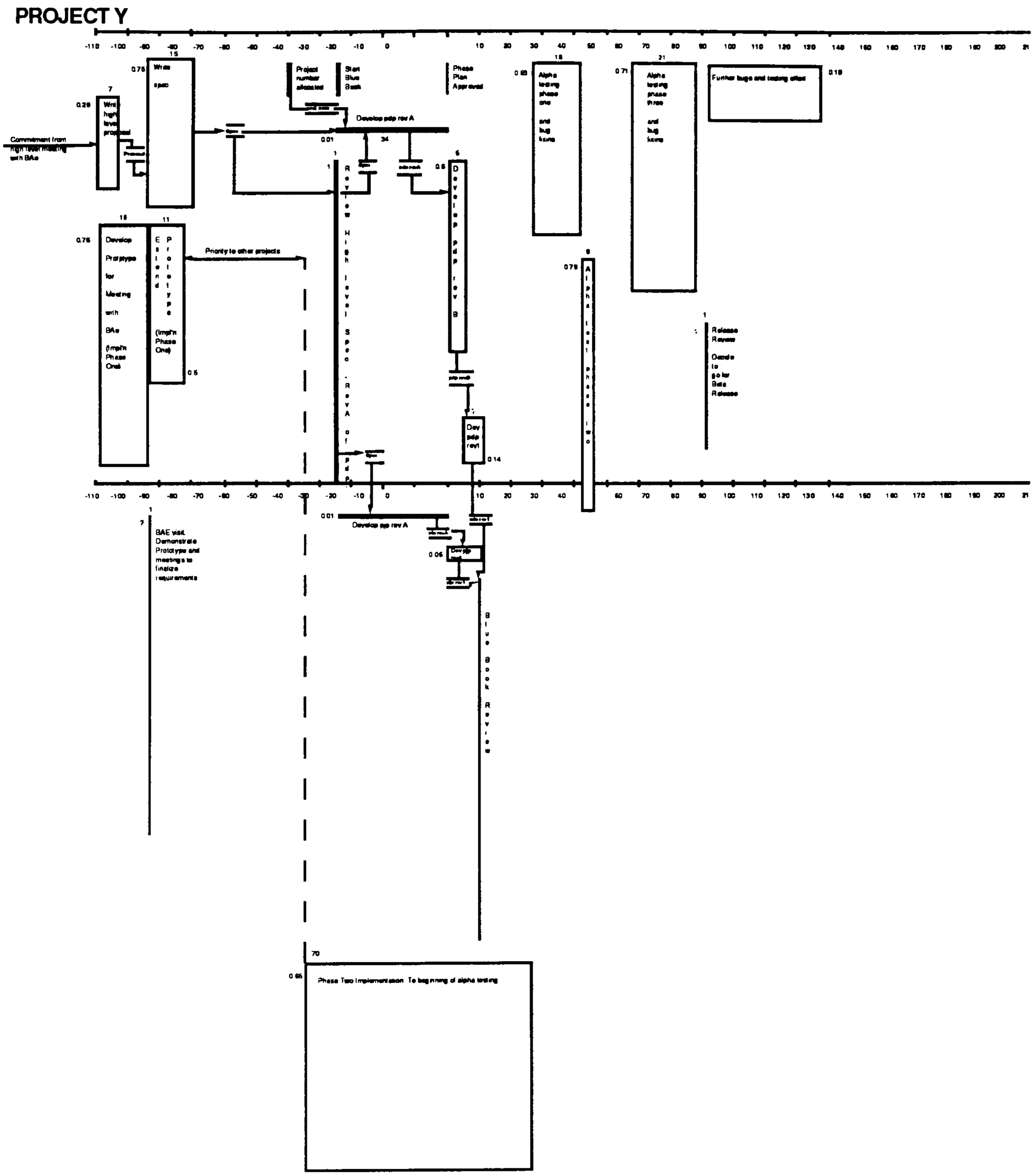

Horizontal : Each unit = 1 day (actual time axis).
Vertical : Each unit = 0.01 people per day (only a scale).
6/9/92
Raise Phase Plan
22
9
0.36
Develop Product Proposal
Revision A
5/8/92
30
17/9/92
0.64
Develop pdp rev C
pdp revB
pdp revC
Develop pdp Rev B
11
Develop pdp Rev D
10
pdp revA
pdp revD
0.39
0.12
Develop Project Plan revision A
32
0.06
Develop pdp rev E
18
pdp revE
0.05
Dev pjp rev B
22
pjp revA
0.02
Dev pjp rev C
23
pjp revB
pjp revC
-30
-20
-10
Day 0
Official project start
10
20
30
40
50
60
70
80
90
100
110
120
For 100% ie 1:1 Print 1 unit = 2mm
For 50% 1 unit (eg 1 day) = 1mm
For 25% 2 units (eg 2 days) = 1mm
5 days or 1 week
?
Phase 1 Implementation
29
1
2
Phase One Review Product Proposal Rev E and Project Plan Rev C
Dev pjp rev1
1
4.13
1
1.5
1
1
Follow on Reviews 20/1/93
pdp rev1
Implementation Phase
Review of Product Proposal Rev F and Project Plan Rev D
pdp rev1
Develop pdp Rev 1
pdp rev1
pdp revF
pdp revF
pdp revE
0.24
Develop pdp Rev F
10
18/12/92
pjp revC
Develop pjp revD
0.08
10

Project Y

The most striking aspect of the model of this project is that there was so much activity before the official project launch (day 0); over 83% of official launch activity and over 89% of all of the activity over the launch period. (That is of all activities during the launch period, even those, which like implementation - which makes up the bulk of the effort - are not considered launch activities, 89% of the total effort over this period comes before the official project launch). Another striking aspect of this project was that the product proposal (which should evaluate the technical feasibility as well as contain requirements) and the project plan each used only 0.2% of the launch activity effort. The justification for this was that the project took a far more prototyping approach, and indeed one might thus expect the nature of the model to be very different. However, in this case the process instance model itself also gives an indication of the success (or otherwise) of the project which had three distinct alpha test phases and yet still required substantial bug fixing and testing effort even after this stage. With hindsight the project manager felt that a major factor in failure of the project was that the initial design was never going to give adequate system performance, and that this issue had been clouded by the wish to produce prototypes in a short time scale (prototypes which had subsequently formed the basis of the project).

The success factors table below, shows that the project suffered from many unexpected problems, much rework, and many bugs. In addition, the lack of customer satisfaction can be further gauged by the fact that at the time of our study the customer was said to not be using the delivered, and still problematic, product.

Part of Project Y Instance Model: Scale 25%



Projects V, W, X and Y were all of similar size, and could have been expected to progress with a similar amount of success. Of the four only project V represents a significantly different type of project, being a software release, whereas the others can be viewed as individual software projects. However, it worth noting again that all of the projects were expected to use the same process.

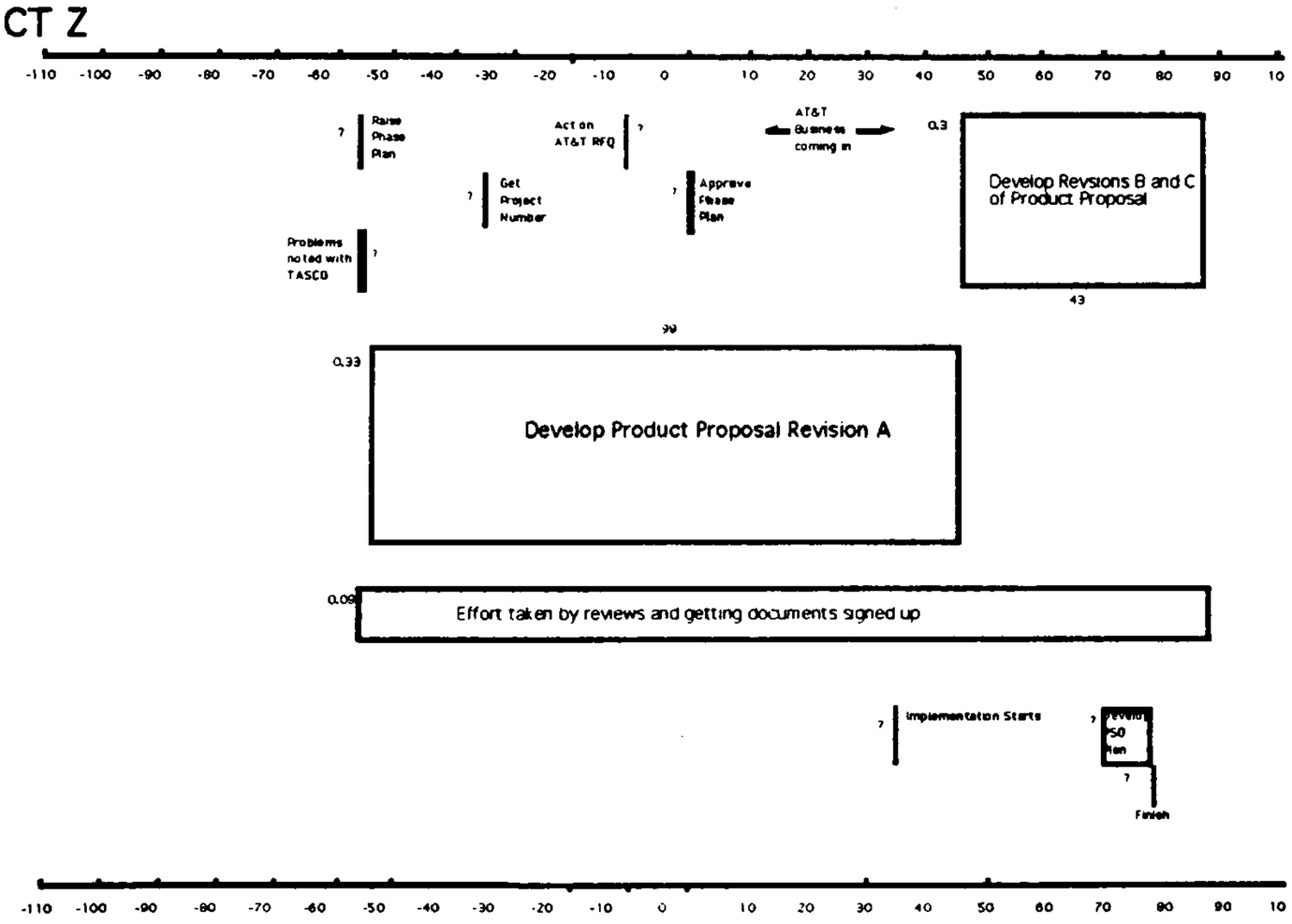
Project Z

This project differed from the other four in both its scale, importance (to the business) and complexity. In addition, the project encompassed both hardware and software design and implementation. An overall project manager (from hardware) was the main driving force behind the project with another project manager being responsible for the software components.

The size, difficulty and importance of the project led to it having high visibility at the site and being resourced by staff who were regarded as among the best. The lack of activities on the model reflects both the genuine emphasis during this project launch on the product proposal, and the fact that it was difficult for us to collect data (hence the single figure for both versions B and C of the product proposal). The enormous effort on the first version of the product proposal suggests that such activity may be even more important for a large complex project than for a smaller one.

The success factors below indicate a satisfactory project. However, this does not adequately reflect the success of this project which was regarded as an important outcome to the organization as a whole (not least because it penetrated a new business area[38]).

Project Z Instance Model: Scale 25%



38 This lack of experience in the product area or lack of 'application domain knowledge' had been one of the reason for the added difficulty of the project. See our remarks in chapter two about how our research confirms some of the findings of Curtis et al. [9, 183, 192-194].

6.3 What's Unique

In this section we concentrate on examining how the novel aspects of our TRADE notation help us to uncover patterns and insights which can be seen in the above models. These insights are due to the unique combination of having a notation with a time-scale, which also shows the project activities, and their effort and duration.

1) Depiction of projects as they occur in time.

The use of a project time-scale provides us with a number of advantages. The official start of a project is the raising and subsequent approval of a phase plan. By classifying phase plan approval as day 0 we can easily see the project activity which precedes this. For example, in project X we see that there is relatively little project activity before this date, whereas for project Y we note that the project is already well under way by this point. We can also look at where types of activity occur in the project, e.g. we can see that the point at which implementation begins varies (compare project X with project Y)

2) Patterns of Activity

We can also examine the pattern of activities over time. For example, whether they are crowded together, whether they occur sequentially or concurrently, whether the activities themselves have had long or short durations in relation to the effort expended on them, whether there is most effort in the initial or latter phases of the project launch and so on. This can also give us much insight into the way that the projects progressed, specifically to the priority or urgency with which they were tackled (compare project W with project X), and where in the launch there was the most effort deployed.

3) Sequencing and parallelism of activities.

The placing of activities on a project time-scale also allows us to see to what extent project launch activities overlap in time. We note that some projects appear to proceed almost sequentially while others have much more concurrency in the distribution of the same activities. In addition, because we still have the inputs and outputs on our model we can investigate which activities are dependent upon the outputs of others.

4) Iterations of activities.

Many notations (e.g. Role Activity Diagrams [244]) allow for iteration, but do not show the extent to which it occurs. However, the extent of iteration is very clear from our models, for example, one project had only one version of the product proposal (project W) whereas another had seven (project V). In addition, because we have effort and duration dimensions we not only know the frequency of iteration but we also know the extent to which certain activity types are revisited, and the respective efforts and time involved in each revisit.

5) Allocation of resources (effort) to activities.

The same activities may have both very different total resource usage, or shape of resource usage (for example, the same total effort might have

consisted of lot of activity over a short period or a lower resource usage over a longer period) across projects[39].

This allocation of effort, and its impact will be examined in our analysis in the following chapter. However, we can see that by combining effort and duration measures with the representation of the activity on the process model we provide an understandable and accessible picture of a project which enables managers to actually see the patterns of resource usage.

This fine grain project detail is representative of the project managers' and engineers' view of their process. Furthermore, these are the kinds of issue with which project managers have concerns, and are able to exercise some control. Having used our notation to describe and illustrate how projects are different, the following chapter includes some discussion, and analysis to enable us to uncover the effects and impacts of some of those differences.

6.3.1 Uniqueness of the Notation

Our notation is unique because it depicts the extent of project deviations. Other notations allow for iteration or concurrency, but they do not depict the extent of these factors. For example, a Role Activity Diagram [244], includes a looping construct so that we can iterate through the same activity. However, it does not depict the number of iterations, nor the differences in effort at each visit. Similarly a data flow diagram allows for activities to proceed either sequentially or concurrently. However, whether activities actually occur in sequence or in parallel cannot be seen. The TRADE notation is much more about depiction or description than prescription. Not only do we see how many times a certain activity type was revisited, but when this occurred, what other activities were occurring at the same time, and how much effort was being spent on each activity.

We will revisit the theme of the impact of effort spent on activity in the following chapter. However, it is clear that our approach is unique in combining a measure of process (in our case effort) in a pictorial representation of that process.

---

39 The product proposal on project X had 36 times greater mean resource usage than that for project Y. In other words project X demanded much more effort on a daily basis, and would thus have a far greater impact on the organization.

7. Interpretation and Discussion of Findings

Chapter Synopsis

The previous chapter described the results of our process modelling work. In this chapter we present an analysis of our results, and suggest some findings and implications of this analysis. We then present our conclusions, suggesting future work, and noting how our work fits within the context of software engineering research to date.

7.1 Analysis

For our analysis we will re-visit the hypotheses which we have postulated for our work (see Chapter Four and Chapter Five) and consider whether our results support or refute these hypotheses.

Hypotheses

Our hypotheses can be considered in two categories, those which are hypotheses about the process modelling undertaken, and those which are hypotheses about the area of process studied (the launch process).

Process Modelling Hypotheses:

H1) Process modelling using simple notations can increase understanding of the launch phase of the development process.

$Treatment_0$:	Baseline. Existing process description.
$Treatment_1$:	Process modelling using data flow diagrams
Response variables:	Discrepancies and problems with existing process. Qualitative. Discussions and feedback from process users about the level of understanding with the existing process description and with the data flow models.
State Variables:	Process modelling (using DFD) applied or not applied.
Effect Investigating	Understanding or insight.

We noted in Chapter Four that we are investigating something (understanding and insight) which is measured as a perception. The only indicators of this are the problems found by using the modelling notation, and discussions with process users.

H2) Process modelling using the TRADE notation can provide insights about the launch process which cannot be gained using data flow diagrams or without process modelling.

$Treatment_0$:	Baseline. Existing process description.
$Treatment_1$:	Data Flow Diagrams.
$Treatment_2$:	TRADE models.

Response Variables:	Qualitative comparison of the kinds of insights gained using the existing descriptions, data flow and TRADE.
State Variables:	The process modelling method used (i.e. no modelling, DFD and TRADE).
Effect Investigating	TRADE provides a richer picture than the other approaches.

H3) Process modelling using TRADE can identify key areas of projects that cannot be determined via DFDs or without process modelling.

$Treatment_0$:	Baseline. Existing process description.
$Treatment_1$:	Data Flow Diagrams.
$Treatment_2$:	TRADE models.
Response Variables:	Ability to identify key tasks that can be used to predict project success.
State Variables:	TRADE: Task effort. Task ordering. Task start and finish (in project days). Average resource usage. TRADE and DFD: Tasks. Inputs and outputs. Discussions with project managers and process users.
Effect Investigating	TRADE provides a more complete picture of process than the other approaches, and allows us to identify key areas of projects.

Launch Process Hypotheses

H4) Effort spent of key launch activities will be a factor in project success.

Treatments (T_1->T_5):	Differing distributions of effort spent on tasks in the launch process of the projects.
Response Variable:	Subjective scores of project success.
State Variables:	Task effort.
Effect Investigating	Impact of effort spent on process launch activities.

H5) The relative amount of effort deployed prior to 'official' project launch has an impact upon project success.

Treatments (T_1->T_5):	Differing distributions of effort before and after 'official' project launch.
Response Variable:	Subjective scores of project success.
State Variables:	Effort before launch. Effort after launch.
Effect Investigating	That pre-empting the official launch of the project has an impact upon project success.

We now consider each of these hypotheses in turn.

7.1.1 Process modelling using simple notations can increase understanding of (provide insights about) the launch phase of the development process.

Our exploratory study succeeded in identifying discrepancies between the documented and actual processes, and in identifying problems with the existing process. We have noted (see section 4.4.1) how the data flow models:

1) Made process understanding more visible.
2) Provided a basis for discussion of the launch process.
3) Gave a coherence to procedures.

Process users discussed (and validated) the data flow models of the launch process. The models were also used in discussion of process problems, and to discuss possible solutions and future process changes. Finally the models made clear the links between a number of process areas - links which had been unclear in the existing process description.

In addition, our presentation of findings highlighted key areas of weakness with the existing process description, and enhanced the feeling of process ownership among its participants. Furthermore, our discussions with those participants (and other process users) revealed that they felt that our work had increased the understanding of the launch process at the site (see 4.4.2). This supports our first hypothesis.

7.1.2 Process modelling using the TRADE notation can provide insights about the launch process which cannot be gained using data flow diagrams or no process modelling.

In section 6.3 we noted how our TRADE notation has allowed us to identify a number of differences among the five projects studied. These project differences include: the amount of activity spent before the official project launch, the amount (and distribution) of effort spent on product proposals, the number of iterations (or versions) of the same document type, and the amount of concurrency among project launch activities. These are project characteristics which we could not identify using standard data flow diagrams (see 6.1) or without process modelling.

The effect of being able to characterize projects in this way, is that we gain further insight into the process under scrutiny. For example, one of the most surprising findings of our study was just how much projects differed at the same site (section 6.3). We have shown that the nature of this project deviation is in the different resource allocations (effort, and duration), sequencing and iteration of project launch activities. This significant project deviation, alone, is a hitherto unknown finding which we directly attribute to the use of the TRADE notation. Hence, our findings support our hypothesis that the TRADE notation provides unique insight into the launch process.

Indeed, the significance of this finding leads us to speculate that, given that TRADE is a previously unused notation, non-conformance is more wide-spread than has previously been appreciated, and that it may be that we have not had the technology to detect it in the past.

7.1.3 Process modelling using TRADE can identify key areas of projects that cannot be determined via DFDs or no process modelling.

We define a 'key area' of a project launch, as some identifiable characteristic of that project launch which may have an impact upon the success of the project. Hence, key areas of projects are potential success predictors.

In informal discussions with staff at the study site (based on the TRADE models of the projects studied) three possible key areas were suggested. The characteristics suggested as key areas were: the amount of effort spent of product proposals, the amount of effort spent on project plans and the amount of effort spent prior to the official project launch. In order to show how our project characteristics (as depicted in the TRADE models) may be tested for their affect on projects we have chosen to take those project characteristics and quantify them. Therefore, we calculate the following measures:

1) Percentage of launch effort spent on producing the first revision of the product proposal against other launch process activities.

2) Percentage of launch effort spent on producing the first revision of the project plan against other launch process activities.

These percentage measures were originally calculated for all activities, however, we have chosen to focus on the first revisions because they are the only definite commonality, that is for each project there will always be one or more versions of both the product proposal and the project plan.

3) Percentage of effort expended before the official project start (day 0).

We conjecture that these measures indicate factors which have an affect upon project success. However, testing this conjecture forms part of our first launch process hypothesis (H4).

7.1.4 Effort spent of key launch activities will be a factor in project success.
and
7.1.5 The relative amount of effort deployed prior to 'official' project launch has an impact upon project success.

In order to test these two hypotheses we have calculated the following measures - as outlined above (in H3) - for all five projects.

Measures to characterize Projects

	Measures	Projects				
		V	W	X	Y	Z
1	% effort on product proposal A	7.39	18.18	25.16	0.20	30.09
2	% effort on project plan	1.85	20.45	8.39	0.20	6.02
3	% of effort before day 0.	12.90	24.09	23.93	83.27	21.02

We have then correlated these measures against our median success score for each project. (The success scores for each project are shown below).

Success scores for instance projects

Characteristic	V	W	X	Y	Z	Median Response
Budget	5	8	8	1	8	8
Customer satisfaction	7	6	7	2	8	7
Forced changes to design	2	5	7	2	6	5
Keeping to specification	4	10	8	9	10	9
Management of Risks	6	5	6	4	7	6
Panics	8	7	7	7	5	7
Post-integration bugs	3	9	7	1	9	7
Re-work	3	7	7	1	8	7
Requirements problems	5	10	9	9	5	9
Within schedule	5	2	7	7	8	7
Unexpected problems	3	7	7	2	7	7
Unplanned over-time	3	10	5	5	3	5
Median Score	4.5	7	7	3	7.5	7

As the table below shows, we have found a significant positive relationship (at the 5% significance level) between the percentage of time spent on the first revision of

the product proposal and the overall success (median success score)[40] of the project.

Project Factors	Overall Project Success
% Project Effort spent on	0.9747
first version of product proposal	sig .0142
% Project Effort on	0.6669
Project Planning	sig .109
% Effort prior to official	-0.513
project start	sig .467

Hence, we believe that our results support the hypothesis (H4) that effort spent on key project activities (in our case the production of the first version of the product proposal) has an effect upon project success. However, our results do not enable us to support our hypothesis (H5) that effort deployed prior to 'official' project launch has an impact upon project success.

We do note that our reliance upon subjective scores (to measure project success) weakens the strength of this result. However, we can use the result to persuade process users of the worth of this activity. Hence, we can use our result, and our process modelling to help to guide the launch process.

7.1.6 A note on the use of hypotheses to structure our analysis

Note that our work was in fact more exploratory than this framework of hypotheses, analysis and conclusions suggests. However, we have found that this framework provides a useful, and hopefully clear, structure for the presentation of our work, such that it might be better understood (and that it might be possible for it to be replicated) by the reader. In adopting such a framework we have again drawn much from the DESMET 'Case Study Design and Analysis Procedures (CSDA)' [228].

We again draw an analogy with the software process itself and the advice of Parnas and Clements [245], who advocate that we should document a project as if it followed a rational process, even if in reality the process has been somewhat more ad-hoc.

7.2 Limitations of our Work

Our most recent research has been limited by the scale of the empirical work carried out. For example, we cannot make strong statistical implications on the basis of only five software projects. Such statistical inference was not our major goal, indeed our use of subjective scores would weaken such an argument. However, we have used our quantitative data to uncover a hypothesis for the organization: that spending time on the first version of the product proposal has a positive affect on project success. We now consider how representative the study site is of software development more generally, and then to what extent we may say that the projects studied are

---

[40] Note the use of medians rather than means due to the fact that we are dealing with ordinal scale measurement.

representative of those at the site, and representative of projects within a wider software development context. Having done this we will then re-examine the extent to which our findings may be generalized.

7.2.1 The Study Site

We described the study site, and the factors which led to our study, in earlier chapters (notably Chapter Four). We now revisit these themes in order to consider whether we have investigated projects at a typical software development site. We identify those factors which are unique to our site and which can be distinguished from those likely to be at work in any software development organization.

The organization which we studied, has a number of sites within the UK, Europe and the US. The site which we studied manufactures a range of products, and produces the hardware and software for those products. We were mainly concerned with the engineering division at this site, who are responsible for hardware and software projects. (Sometimes they may produce a joint software and hardware solution within the same project). All projects are within the same product application domain, though there are a variety of customers, and these customers span different business domains. Projects may vary in size. Some projects are to upgrade existing products, whereas some are to produce new products (within an existing product range). All procedures - which projects are meant to follow - (including launch procedures) have been defined and the organization is TickIT accredited [59]. (This accreditation is a theme to which we will return in examining the implications of our findings).

We chose to approach the specific site, because we had some previous, and favourable, experience of working with them. This allowed us good access to staff and data at the site. An added bonus was the geographical proximity of the site, which meant that visits could be frequent and relatively easy to arrange.

We were not able to control project factors, such as personnel, complexity, and size. Hence, we chose to pick a representative sample of typical past projects (see section 5.2.6). We will now examine a number of project factors. These relate both to the nature of software development at the site, e.g. procedures, and methods, and to specific project factors such as size or complexity. We will then use this discussion to consider the extent to which the projects studied may be considered typical of software development.

7.2.2 How Typical are the Software Projects

This section considers the extent to which the projects we have examined are typical. There are a number of levels at which we will examine whether the projects studied are typical: of the site, of real time software projects and of software development projects in general. The following section will then use these arguments to consider which aspects of our work are generalizable; to the site, to real-time development and to software development in general.

7.2.2.1 Typical Projects at the Site

There are a number of factors by which we can characterize projects at the study site. We will now consider these factors, and the way these factors are represented by the five projects studied. We will then argue that the projects studied are typical of projects at the study site.

Application Domain

All projects at the site are within the same application domain. Though customers may be from various business domains, they are all being supplied with essentially the same kind of application.

Languages

The programming language used by the majority of projects at the site is C. The user interfaces are built using an 'in-house' graphical language which interfaces to C. Of the projects we examined all were primarily written in C, which is typical of projects at the site.

Enhancement vs. New Products

The majority of projects at the site are to make enhancements to existing products or product lines, or to include enhancements in the next product release. These projects are of the kind represented by projects V, W, X, Y.

There are also some projects which represent a departure from this pattern, for example, to venture into a new business area, or to work with a new and potentially important customer (e.g. project Z).

Size

Projects may vary in size. Most projects are of the size of V, W, X, Y. However, occasionally there is a much larger project which may incorporate smaller sub-projects (e.g. project Z).

Compression of Time-Scale / Schedule and Pressure or Urgency

We have considered projects to be of a similar size if they require similar amounts of effort. However, the launch phase of those similarly sized projects may occur over differing periods of time. For example, project W and Project V expended similar effort over the launch phase but project W took much longer to do it. This may give some insight into the urgency or priority of individual projects.

In order to represent this difference in urgency we have included projects which do appear to have these different effort distributions (and thus we are being more representative of projects at the site). Secondly, these are exactly the kind of issues which are made more visible by the use of our TRADE notation.

Our five projects examined covered the spectrum of urgency, from projects which were fairly low priority (W) to those which were extremely urgent (X and Y), and those in-between (Z, V). Furthermore, we found no obvious link between urgency and success.

Staff Experience

We have noted (Chapter Two) the arguments that staff are the biggest productivity factor in software development. Hence, it could be argued that the variation in staff expertise should outweigh any other project success factors. We have already considered these arguments in our discussion of confounding factors for projects (Chapter Four and Chapter Five). However, we also need to consider whether the allocation of staff (or expertise) on the projects examined is typical. For example, we need to consider whether the projects which we have examined were those with the keen staff, or the domain experts, or those who care about process and quality.

In interviewing staff across the projects we encountered a wide variety of attitudes. We talked to staff who were quite negative about process and those who were enthusiastic. We talked to some who were on projects where they had a clear picture of the project and the domain (including one person whom Curtis would describe as a super-designer [196]) and those who felt that they had little project or product understanding.

Moreover the projects were specifically selected by the organization to be a representative mixture of typical projects. The organization was keen that we examined as representative a sample of their projects as possible, and that this should include both successful and unsuccessful projects. This selection was not made by us but by the engineering director and the quality manager at the site. These people have an overview and understanding of the projects at the site, but no direct allegiance to any specific projects. Furthermore, they gave no indications (either to us or to those involved in the study) of their opinions about the relative merits of the projects under scrutiny.

Project Manager

For the production of our instance models the majority of information came from the project manager. (Note that we considered that the project manager would probably be the biggest staff influence on success. However, two projects with very differing successes were managed by the same person - see Chapter Five).

There may be some bias here, in that we did not wish to examine projects whose project managers saw no value in the exercise, and who did not wish to participate. If we had done this then we would have very little faith in the data collected from these projects. In theory this constraint means that we could not look at projects where the project manager was very negative about process. This is a potential weakness in our arguments about the projects being typical. For example, it might be argued that project managers who are hostile are also those who have a different project process to the ones we examined. Though we believe that this is unlikely, we cannot extend our arguments about our projects being typical to those where the project manager is hostile to our modelling technique. However, we believe that for neutral (as in projects V, W and Z) or co-operative project managers (as in projects X and Y) our results are valid.

Project Difficulty

We examined one project (Z) which was both large and very difficult. The other four projects were all considered fairly typical of those undertaken at the site.

Again these are views which we subsequently (post study) learned from the engineering director and the quality manager, and which agree with the views expressed by the staff interviewed during the course of our study.

Project Process

All projects examined were to follow the same defined launch process. Though we have suggested (after modelling with TRADE) that there were considerable deviations among the projects examined, this would not be apparent from procedures, nor would it have been apparent from modelling using data flow diagrams. This is the procedure followed by almost all projects at the site. The only projects at the site which did not have to follow this procedure were those where an important customer asked for a particular small one-off enhancement. These projects known as 'specials' were not covered by the launch procedure, and were not within the control of the engineering group, hence, we do not extend our findings about success predictors to include them.

Summary

We believe that the projects which we have examined are typical of the projects at the study site. Our only real concern is that we only examined projects where the project manager agreed to take part in the activity. Therefore, we believe that our findings are generalizable to all (non-specials) projects at the site with neutral or supportive project managers[41].

7.2.2.2 Typical Real-Time Software Projects

We now examine project and organizational factors which may challenge the representativeness of the projects studied within the wider context of real-time software development. We first revisit some of the factors examined above and then suggest some additional factors which may have an impact upon the project process. Note that for projects other than those at the study site (see above) we are only considering generalizing with respect to our findings on the extent of project deviation and not our more specific findings on the impact of the product proposal document (see 7.2.3.1).

Application Domain

We have examined projects (at the site) which do not differ in application domain. Clearly among real-time projects in general there will be a number of different application domains. However, we have investigated a single domain and still found extensive project variation. We might, therefore, conjecture that since we have investigated a site with the least possible variation in application domain, sites with projects across many domains may exhibit similar or even greater project variation. (We note that Curtis et al. [196] suggest that application domain understanding is the most important factor in programmer productivity. Hence, we can conjecture that a number of projects involving new domains may have significant process consequences). However, this is only speculation. A number of other factors may influence the project process (as outlined below).

Languages and Methods

A number of languages, and software methods may be used within real-time software engineering. Indeed, some sites, or even some projects may utilize more than one method or language. Therefore, since we only examined projects which were primarily written in C we cannot claim that our projects are typical of all real-time development.

Methods and Tools

At the site we investigated there was no common development method (e.g. Yourdon [202]) across all projects. The use of a specific design method across an organization (particularly if supported by CASE tools) and the shared use of a common design terminology may lead to greater process conformance than we discovered in our investigation.

Enhancement vs. New Products

---

41 This means any project managers who are not hostile to the modelling technique or to the data collection exercise.

At the site we investigated the majority of projects (and the majority of overall development effort) involved maintenance or enhancement of existing products. At other sites there may not be the same mixture of projects, or proportion of development devoted to maintenance, evolution, enhancements or new products. This is another factor which may influence the development process. For example, we might expect a process which catered for a diverse range of new products across a number of application domains to need greater flexibility than one which catered only for enhancements to a small existing product range. Though we have no evidence to suggest whether such expectations are reasonable, this is an area which would need further investigation if we were to attempt to generalize our results to real-time software projects.

Project Size and Complexity

Though projects at the study site varied in size, (from three or four people involved over the project life-cycle to twenty people) all projects were within a relatively small engineering group (of approximately 50). Within real-time software development projects may involve far more people, over greater periods of time, and at much greater cost. For example, Rifkin and Cox [246] describe one case as follows: 'This is a major DoD project in the defense systems arm of a major aerospace firm. At award it was estimated that the project would take 38 months to develop approximately 325,000 source lines of Ada. About 75 professionals are assigned to the project.'

Some organizations will exhibit much greater variation in the size, duration and complexity of their projects than the one which we have studied, whereas others may have a high number of very similar projects. Clearly the extent to which the process users understand and feel comfortable with their process will be influenced by the variety (and different natures) of the projects the organization undertakes. For example, if all projects involve three or four people for a few months, doing the same kind of work, then we might expect there to be greater conformance. Again we do not know whether this is the case or not, however, it is a possible factor which would have to be investigated if we were to attempt to generalize our result about project variation beyond the study site.

Compression of Time-Scale / Schedule and Pressure or Urgency

The time-pressure under which process users are working is another possible factor which might effect process variation. For example, process users might be tempted to 'cut corners' under severe time-pressure. This is an effect which could be investigated both at the project level and at the organizational level. For example, the extent to which process is adhered to under pressure may vary among different organizations. Further work would be required in order to investigate which factors influence the extent to which users keep to the process.

Staff Experience and Expertise

Basili and Hutchens [195] have suggested that the performance of individuals is often the greatest factor in project success. The view that the experience or expertise of staff on a project may have a significant impact is also held by other authors (e.g. Curtis et al. [7]). Hence, it is not unreasonable to suppose that staffing may also have a significant impact upon the project process. We have discussed this issue for the studies carried out at our site, noting in particular the likely impact of the project manager. Within the larger context of real-time development we cannot dismiss the possibility that staff may be the biggest impact on the success of projects.

If staff are the factor which has greatest impact upon software projects then the recruitment strategies of different organizations (which may lead to a different make-up of staff) may also have an influence on the project process. For example, there may be a strong emphasis on choosing project managers with particular qualities.

In addition, the extent to which process users are educated about the processes within an organization, may have an impact upon process understanding and process conformance. For example, at the site we studied there was a standard Software Engineering Training Programme (SETP) to educate all process users. Hence, staffing and the experience of staff at the site may have a significant impact upon the process, and upon the project process variation within a site.

Project Process / Processes

Clearly the procedures and processes used at other organizations will be different to those which we have investigated. For example, one might expect that projects in an SEI level 3 organization would exhibit less variation than those in an SEI level 1 organization (see also our comments on process assessment and process quality initiatives below). However, again this is an area which would require further empirical study.

Furthermore, the methods of documenting and presenting those procedures to process users (see our comments on process education above) may differ. This again has a possible impact upon process conformance.

Process Assessment and Process Quality Initiatives

The introduction of assessment initiatives (see Chapter Two) may increase the focus upon process within organizations. Certainly this was the case within our study site. One might expect that such an increase in attention would lead to better process definition, and increased process conformance. However, once again we believe that this is an area that deserves further study (see our arguments about quality initiatives, assessment and accreditation in section 7.3.1).

Summary

Different organizations or sites will have differing processes, educational and recruitment strategies, quality strategies and application domains, all of which will make them distinct and different from the site which we have investigated. However, the range and diversity of projects at any particular site is perhaps the biggest threat to the representativeness of our study. Though we have no reason to believe that our projects are atypical, we cannot state that our projects are representative of real-time software development projects in general.

Nevertheless we feel that significant variation among projects may also exist at other[42] real-time sites, and that some of the factors discussed above may even contribute to greater variation. For example, a site with a great diversity of projects, covering multiple application domains, of a similar process maturity to the one we investigated might well have even greater diversity among the projects. However, we might also expect that a site with a number of small similar projects, with more thorough process education, and higher process maturity might not exhibit the variation in projects which we discovered at the study site. In summary we would argue that our results suggest the need for further work, in order to investigate whether similar project variation exists at other sites.

---

[42] That is, other than the site we studied.

7.2.2.3 Software Projects in General

If we attempt to examine whether the projects studied are typical of software development in general we will again need to consider a number of additional threats to their representativeness. For example, the range of application domains is greatly increased, as are the languages and methods used for development.

Indeed, the scope of projects is so varied that we do not wish to suggest that our findings can be generalized in this way. However, once again we suggest the need for further empirical work in order to investigate the extent of project deviation in software development.

7.2.3 How Generalizable are the Project Results?

We have argued that we have studied typical projects at the site, and since we believe that our launch process hypothesis (H4) is plausible[43], we believe that this result (of H4) is generalizable across the (non - special) projects at the site which do not have hostile project managers. Therefore, it is our logical inference[44] that spending time on the first version of the product proposal will have a positive affect on projects at the study site. However, we note that this is only a preliminary result. We can only say that our preliminary results support the hypothesis that time spent on the first version of the product proposal has a positive impact on project success. We would need to conduct a much larger study in order to verify this generalization statistically. Furthermore, this is only a preliminary finding for the site at which we carried out our studies. Other sites, and other organizations will have different processes, and different process activities, and we cannot generalize our result to them. Nevertheless it is gratifying to find some empirical evidence which appears to support software engineering theory (that spending time on early requirements activities is cost-effective).

Our most general finding concerns the nature and extent of project deviation: briefly that project deviation is far greater than we had expected, given that all projects follow the same defined process. We believe that this is a finding which will hold for other projects at this site. Indeed, we would expect that such deviation is even more likely among those (special) projects which do not follow the launch process. However, once again we would need a larger study in order to verify this assertion.

Whether our finding on project deviation will hold for other similar organizations remains to be evaluated. Rather than suggest that our results are compelling enough to imply that all organizations will have projects which exhibit such deviation, we see the role of our study as providing a counter-example to the established orthodoxy. We believe that there is a need for further work of this nature, within different organizations and domains. Such work might of course suggest that our findings are atypical. However, we believe that the use of similar methods and notation would yield similar results, and that a number of these studies together could form the basis of a generalizable result.

---

[43] Hammersly [221] argues that it is the plausibility of hypotheses, not the strength of statistical argument that ultimately leads to their acceptance. That is, that we will not accept statistical inference alone, but that this must be accompanied by a plausible hypothesis.

[44] Hammersly [221] further argues that we should accept or reject hypotheses based on the strength of the logical (not the statistical) inference.

7.2.3.1 A Summary of the Generalizability of our Findings

We do not believe that the same practices e.g. spending time on the product proposal (or some equivalent document) is a result which will necessarily hold outside our study site. Indeed, it is this very point, that organizations need to investigate what are the successful process activities for their own site, and which are specific to their site, that motivates our process modelling efforts.

However, a similar study to ours, using the TRADE notation, at a similar site might also yield insights about the nature of their development process. In addition, the extent of process variation among projects might again be substantial.

7.3 Implications

In analyzing our hypotheses (see section 7.1.1.2) we noted that our use of the TRADE notation has allowed us to uncover significant deviation among the projects studied. We discovered that there was far less process commonality than had been assumed. Since all projects were supposed to follow the same defined (and TickIT accredited [59]) process, this brings into question just how much accreditation influences process conformance. Furthermore, our preliminary findings suggest that the kind of deviations uncovered may have an effect upon project success (7.1.1.4). In other words these 'key differences' among projects are not made visible by the TickIT certification process.

This is such a potentially disturbing implication of our work that we will consider it in some depth. We first consider the questions which our study raises. We then revisit some of the ideas behind quality initiatives (notably TickIT), and we then consider some possible explanations for our findings.

We also note parenthetically that given that this kind of project variation is not captured by current modelling methods it also brings into question whether such methods are appropriate for project control (see 7.3.3).

7.3.1 Implications for Process Improvement and Quality Initiatives.

Our project variation findings lead us to consider a number of questions.

1) Is this kind of project deviation common or is our site atypical?

   We have used a new notation at our study site. It may be that our site is typical and that such deviation is widespread, and has been previously undetected because the modelling technology to detect it has not been available. Alternatively, such deviation could be uncommon, and our site could be atypical.

   We have discussed the extent to which we can say that the study site, and the projects modelled are typical in the previous section. We cannot argue that the projects studied are typical of software development outside the study site. Hence, we cannot say whether other sites will exhibit the same kind of deviation among their software projects. Equally, however, we have no reason to think that they will not, particularly since the modelling technology which we used to discover such project deviations (TRADE) is unique. Clearly there is a need for further work of this nature. At present our preliminary findings remain as a counter-example to the established orthodoxy.

2) Does process definition lead to process conformance?

Most of the work to date on software quality assurance seems to assume that process conformance can be achieved by having a defined quality system, and then attempting to assure adherence to that system. Indeed, within the UK the well publicized BS5750 and TickIT [59][45] accreditation schemes lay much emphasis on process definition. Hence, there is an assumption within TickIT that process definition will lead to process conformance.

Our work brings into question the viability of such process conformance and thus the assumptions on which much of this work is based. We have examined a TickIT accredited [59] organization and found significant differences in software projects. Moreover, these are differences which were not made visible by the certification process. Hence, we might argue that the certification process is in some way incomplete in that it does not provide a rich enough picture of process.

However, it might still be argued that certification and process definition leads to increased process conformance. Our work is unable to refute such a supposition in that we have not examined any 'non TickIT' projects, and are, therefore, unable to compare process conformance with and without certification. Again, this suggests a need for further work.

3) Does process definition and certification give us a better process or lead to better products?

Many quality initiatives appear to be based upon this very premise, that is, that certification will improve not only the software process but also (as a result) the software products produced. However, our work shows that there may be key differences in projects even where there is an accredited process definition. More importantly these are differences which may have an effect on project success[46] and are the kind of concerns (e.g. resource allocation and usage, sequencing of activities) with which project managers are familiar. This suggests that there is more to quality assurance than process definition, and that we may also need to tailor quality initiatives to the organization or site rather than relying solely upon industry 'best practices'[47].

Once again we are unable to state that certification does not lead to a better process, because we have no comparison between certified and non-certified projects. However, our findings do lead us to question whether we can expect that process definition alone will lead us to a better process or to producing a better product.

4) Does certification allow us to discover the key areas of our project processes?

---

[45] Which is essentially the same as the ISO 9000 series.

[46] We note the comments of Curtis et al who speculate about the value of process models which exclude those factors that have most influence upon the process outcomes [9].

[47] Therefore, it may be that we need more emphasis on the role of the local Quality Management System than on the global quality scheme.

In examining our third hypothesis (H3) we proposed that the differences among projects which we had discovered using our TRADE notation might be key areas of the process (see 7.1.1.3). In other words, such project differences might account for project success, and might (with further study) form the basis of some prediction system (see point five below).

Clearly if certification schemes do not identify differences of this nature (certainly they did not at the site which we investigated) then we must question what key process areas they do allow us to uncover.

5) Does certification provide us with any possible mechanisms for predicting the success (or otherwise) of projects?

We note that the process certification schemes used at the study site had not provided or uncovered measures which could be used as project success predictors[48].

7.3.1.1 Background and Discussion of TickIT Certification

We discussed a number of process assessment and process capability evaluation frameworks in Chapter Two (section 2.3.2). Of these, perhaps the most influential has been the Software Engineering Institute's work on Software Process Maturity [147, 148] and the Capability Maturity Model [60, 149].

The use of these assessment frameworks solely for an examination of an organization's processes is only part of the motivation for their use. Another important motivation for their use, is to gain some form of accreditation. We have remarked already (section 2.3.2) that the US Department of Defense wished to use process maturity to be able to assess the capability of contractors. Similarly, organizations themselves, wish to be able to gain accreditation (whether this is a process maturity level, or TickIT, or some other assessment initiative) in order to gain contracts. For example, Davis et al. [247] note that in a survey of reasons for developers seeking or not seeking an external Quality Assurance standard 'only 12% of the respondents with third party certification stated that reduction of development costs and increased productivity were of high consideration'. The same survey gives the main reasons for being 'assessed to an external standard' as:

1) Commercial success.
2) To improve reliability.
3) To produce a system of known quality.
4) To satisfy end-users.

For the organization which we have studied, certification had meant not only examination of processes, but also 'respectability' and competitive advantage. Since the site was in the UK, they had sought (and gained) TickIT accreditation. We thus concentrate our discussion upon TickIT.

Hunter and Rae [248] trace the development of TickIT back to the publication of a report entitled 'Software a Vital Key to UK Competitiveness' [249] by the Advisory Council for Applied Research and Development (ACARD) in 1986. This

---

[48] We contrast this with our own preliminary findings which suggest that time spent on the first version of the product proposal appears to have a positive impact upon project success. Though we would need further work to verify these preliminary results with further investigation of projects we might have the basis of a prediction system. In other words we have discovered possible predictors of success.

report 'found that software quality in the UK was generally low' [248]. As a result of the report the UK government commissioned an investigation into commercial requirements for software standards. Among the recommendations of this investigation were: (1) to attempt to harmonise existing standards (civil and military) and to review the guidance associated with them, and (2) to study the cost and benefits of such standards. These tasks were then carried out by two UK consultancy firms: Logica, who compared existing QMS standards [250], and Price Waterhouse, who examined the cost benefits of such standards [251]. Hunter and Rae state [248] that the subsequent appointment of the British Computer Society (BCS) to lead an initiative called TickIT was 'in response to the findings of both the Logica and Price Waterhouse reports'.

The TickIT initiative was intended to provide a mechanism for marketing, reviewing and updating the BS5750, which is the UK equivalent of the ISO 9000 series. This response was due mainly to the findings by Logica that there was little difference between exiting British civil, military and nuclear standards and ISO 9000 standards [248].

TickIT has sections on general framework, life-cycle activities and supporting activities. Although the existence of a software life-cycle is noted, there is no recommendation of any particular model of the software development life-cycle. However, the traditional life-cycle phases of requirements, design, implementation and testing are treated as separate topics. TickIT also requires the production of specific plans: a development plan, a quality plan, a test plan and a maintenance plan. As with the SEI Capability Maturity model, TickIT assessment can be considered to be merely bench-marking. That is, that assessors are looking for the existence of certain 'best practices' or criteria which are considered to be those things which a software developer should do. For example, there must be a defined software process, and the assessment is also supposed to examine that this process has been adhered to. However, it is not clear exactly what this adherence constitutes, or how it is measured. Furthermore, the extent to which we can judge process adherence depends both on the mechanism used to describe the process, and that used to check that projects conform to that process. Our own work suggests that at the site we studied, the actual deviation among projects was not discovered by the TickIT certification. This is probably due to a combination of the following reasons, namely that:

1) The existing process description was incomplete (for example, it did not include guidance about effort).

2) The process assessment required for certification could not check for these effort criteria since they were not described by the existing procedures.

3) The TickIT certification was more concerned with checking that there was documentation to show adherence (by projects) to the defined processes, than observation of what that process really was.

   Even our exploratory study showed a number of discrepancies between documented and actual process.

4) TickIT does not require the kind of measurements of process which we subsequently used.

5) There was no existing modelling technology which would describe the project processes in such a way as to discover the variation.

TickIT is primarily concerned with the definition and documentation of a quality management system, and thus the associated definition of the software development

process. In addition, certification requires that there is some evidence that these standards are adhered to, that projects conform to the defined process. Clearly even the results of our exploratory study cast doubt upon this method of measuring conformance used at the study site. Furthermore, our later investigation of projects shows that projects actually had far less commonality of process than accreditation might imply.

Avison notes that 'Documentation of the quality system does not necessarily mean understanding and commitment' [252]. Our exploratory work confirms that process understanding requires more than process and quality standards definition and documentation. Furthermore, we suggest that documentation of the development process does not necessarily lead to process conformance either.

Perhaps more disturbingly there is no empirical evidence to suggest that this kind of process definition actually improves the software product. There is no need for a TickIT accreditation to assess anything other than processes. Therefore, a certified process may still produce poor software products (thus it may not improve reliability, or produce a product of known quality as was desired by the respondents of the above mentioned survey [247]). That is, that an accredited organization may not necessarily produce better products than an organization which is not accredited. Furthermore, the same organization might not improve its software product either to gain, or as a result of accreditation.

The proponents of such initiatives might argue that by increasing process standardization or process conformance we will have a less risky process. That is, that we may not always produce good products, but that we are less likely to have chaotic projects. We return to the theme that TickIT (and other similar quality and process assessment initiatives) are based on the assumption that they will help to achieve process conformance. Again, it may be that accreditation does increase conformance, and we have no evidence to suggest that it does not (see our comments in questions 2 and 3 in section 7.3.1). Equally, since we have found significant project variation at an accredited site, we suggest that the extent to which accreditation increases process conformance may be more limited than the proponents of such initiatives might suggest.

7.3.2 Implications for Other Quality Assurance Initiatives

The arguments above about TickIT clearly have implications for other (other than TickIT) process improvement initiatives, notably those which share a view of the importance of process conformance (for example the SEI Capability Maturity Model [153], the ISO 9000 series [60, 62] and SPICE [142]). For example, the significant differences in types of software project even at one site may mean that we need to take a much more site-specific view of process improvement, which is more in line with Koch's idea of 'self-referential process improvement' (see BOOTSTRAP [253]). However, this does not mean that we should abandon ideas about what constitutes good practice, rather that we should try to investigate the effect of these practices within an organization (perhaps by having an instrumented process) before we commit them to policy.

7.3.3 Implications for Process Modelling

We believe that the startling variety among projects which supposedly follow the same well defined process (the one we investigated was TickIT [59] approved) is a finding from our research which also has repercussions for the process modelling community. Specifically we believe that our findings suggest that:

1) Process modellers need to be aware that there may be process variation which is external to their modelling notations.

For example, both the existing launch process description, and the data flow description of the launch process, did not capture the project variation which we discovered using TRADE. We need to be aware that our models are incomplete descriptions of the actual process, and that they may not capture certain kinds of information about the process.

This is particularly important if we wish to use process modelling to control the process in some way. Specifically we need models which capture the kind of process variation which we wish to control.

2) Simple process models may be used in order to understand the process.

Process conformance at the study site was limited not only by the technology which was used to model the process but also by the existing (common) understanding of the launch process. Though our exploratory study notation did not provide us with a rich enough picture of process to discover the variation in projects, it did allow us to gain and promulgate process understanding. Having gained a better understanding of the process at the study site, we were then able to direct our further modelling efforts more effectively.

Using simple process models to gain understanding of the development process, may be one of the important roles of process modelling. This does not mean that we should abandon our attempts to control the software process. On the contrary, there may be some aspects of the development where control is vital[49]. For example, there would be little point in having a configuration control process that was not rigorous. Therefore, it might be appropriate to use simple graphical process models to gain process understanding, and then build more elaborate, more complex or more formal models to control parts of the development process.

3) An important application of process modelling is to help guide the software process.

Process models may be used for discovering and encouraging good practice (providing process guidance).

Many of the process problems and discrepancies which we encountered were traced to a lack of understanding of procedures, and or process, which suggests a need for process guidance. We believe that this process guidance, which we distinguish from process control (see [12, 99, 254, 255] and above), is one of the main roles or benefits of process modelling (particularly when using easily understood graphical models).

We suggest that instead of attempting to impose standards and models which are said to represent best practice, we would be more profitably employed by trying to find out just what it is within our own organizations which leads to project success. Having found this, we can then use process models in an educational role, to show the consequences of spending time

---

[49]We might need to establish what parts of development need rigid control and what do not. A good understanding of the process, and the effects of process activities, would be beneficial in deciding the appropriate level of control for different parts (or aspects) of the development process.

and effort on key process activities. Hence, process modelling can be used to guide the software development process.

There are also some implications for specific branches of process modelling work, notably.

4) The IPSE community and automated process support technology.

These (project support technology) applications of process modelling put great emphasis on implicit strong process control through enacting process models (e.g. [129]), providing a process description which is to be instantiated for each project. However, our preliminary results suggest that project control through adherence to, or instantiation of, some generic process may be far more difficult than has been previously thought. We believe that it may in some cases be more profitable to understand the development process at a site before attempting to impose instances of some generic process model.

7.4 Implications for future process modelling work

We have suggested that our work has some implications for current process modelling work, and for quality assurance initiatives (section 7.3). In addition, we believe that there are some implications for the direction which future process modelling work might take. We first examine these implications, and then suggest how they might influence future process modelling initiatives.

1) Process modelling and software metrics have complementary roles.

One can aid the other, i.e. modelling aids measurement, and measures help to provide more illuminating models.

We confirm the view that process modelling can aid data collection by providing a framework (in our case generic activity types) for software measurement [51, 156]. This helps us to know when to measure, what information we are collecting, and how this data will be used to instrument the process.

2) Having a generic process description can aid data collection.

We have stated that process modelling and software metrics are complementary. However, for our study the process model had two differing objectives, to discover or represent the nature of software projects, and to aid data collection. We used the genuine process invariants to create a generic model to collect data against, though the pictorial representation of projects took a different form. Compare our collection sheet or super-template (the generic model) with our extended data flow notation or instance models (the projects or instances of process).

This generic model is really realized by a single process instance. It is the equivalent of Parnas and Clements' faked 'rational design process' [245], in that it is an idealized process which never really happens. In other words, the generic model can be useful if we realize that it is not an accurate description of reality and simply use it as a tool to help to order our data collection strategy.

3) Pragmatic approaches to process modelling can provide useful results

The success of our modest study suggests that the low technology application of process modelling, particularly in order to increase process understanding, appears to offer good prospects for the acceptance of the technology as a whole.

We again note that other successful process modelling studies have also taken a similarly pragmatic approach [18, 197, 200, 215]

4) There should be greater emphasis on industrial and empirical process modelling work.

We believe that there is a need for further work to consider the practicalities of applying process modelling methods, and that this can only be usefully achieved by industrial collaboration.

In addition, many of the problems encountered in undertaking a process modelling program were to do with people, rather than being of a technical nature [10, 191]. The lessons learned in Chapter Five and Chapter Six will go some way in providing further guidance of a non-technical nature to the potential practitioner, but again there appears to be a pressing need for more reported experience of this kind.

We once more return to the theme of our opening chapter, that there has been much theoretical work in software process modelling, but relatively few empirical studies. Despite some notable exceptions (e.g. [18, 197]), few studies have attempted to transfer process modelling technology to industrial problems. We believe that empirical work should be the priority for process modelling research. In supporting such work we further argue that there is a need for more pragmatic approaches to process modelling. Finally, we suggest that process modelling will be a more powerful tool if it is combined with software measurement. Indeed, our own work has provided further insights into the process under scrutiny by using a technique which combined metrics with process modelling.

7.5 Conclusions

This research has investigated process modelling in practice. An exploratory process modelling study (using data flow diagrams) was undertaken in order to increase understanding of the launch phase of software development at an industrial site. This exploratory study discovered discrepancies between the actual and theoretical (documented) process at the study site. Models produced by the study were then used, in conjunction with a report, as the basis of a presentation to staff at the site, to highlight process problems, to increase process understanding and to suggest possible process changes. Changes to the process documentation were then made based on the recommendations of this exploratory study.

A follow-up study of the launch process examined five projects at that site, utilizing a new process modelling notation (TRADE) which combines effort data with the process model. This new notation provided a richer picture of the launch process and led to new insights about the nature of the launch process. (For example, the amount of effort spent on the first version of the product proposal was markedly different across the five projects examined). The TRADE notation has also enabled the identification of key areas of projects which may be possible predictors of

project success. (Our preliminary findings suggest that project success at the site[50] can be related to effort spent on the first version of the product proposal).

The use of the TRADE notation also led to the observation that the projects studied exhibited significant deviation in their launch processes (deviation which we would not have expected at a TickIT accredited software developer). This observation has led us to re-consider the effectiveness of accreditation at the study site, and brings into question the relationship between process definition, process conformance and quality assurance (see section 7.3) within software development.

Our process modelling study not only complements the existing pragmatic approaches to process modelling, but shows how process modelling can be made more effective by combining data collection and display in a process modelling notation. In addition, the use of this (TRADE) notation to discover significant deviation among the projects at our study site, provides a rare (and disturbing) insight into the nature of process deviation at a (TickIT accredited) software development site.

---

[50] For non-specials projects with a project manager who is not hostile to the use of the TRADE modelling technology and associated data collection.

References

[1] Brooks, F.P., *No Silver Bullet: Essence and Accidents of Software Engineering*. *Computer*, **20**, 4, pp10-19, 1987.

[2] Kitchenham, B., *Making Process Predictions*, in *Software Metrics: A Rigorous Approach by Fenton, N.E.*, Chapman Hall: London, 1991.

[3] Humphrey, W.S., *Software and the Factory Paradigm. Software Engineering Journal*, Sep, pp370-376, 1991.

[4] Moularde, J.-P. and R. Rockwell. *Life-Cycle Industrial Process Practices: Why They Work and Don't Work: Position Statement*. in *Proceedings of the First International Conference on the Software Process: Manufacturing Complex Systems*. Redondo Beach, California, USA: IEEE Computer Society Press, 1991.

[5] Basili, V.R. and J.D. Musa, *The Future Engineering of Software: A Management Perspective. IEEE Computer*, Sep, pp90-96, 1991.

[6] Zave, P. *Domain Understanding and the Software Process*. in *Proceedings of the 5th International Software Process Workshop*. Kennebunkport, Maine, USA: IEEE Computer Society Press, 1989.

[7] Curtis, B., H. Krasner, and N. Iscoe, *A field study of the software design process for large systems. Communications of the ACM*, **31**, 11, pp1268-1287, 1988.

[8] Curtis, B. *Three Problems Overcome with Behavioural Models of the Software Development Process*. in *Proceedings of the 11th International Conference on Software Engineering*. IEEE, 1989.

[9] Curtis, B. and N. Iscoe, *Modeling the software design process*, in *Empirical Foundations of Information and Software Science V*, Editor(s), P. Zunde and D. Hocking, Plenum Press: 1990.

[10] Rodden, T., *et al. Process Modelling and Development Practice*. in *Proceedings of the Third European Workshop on Software Process Technology*. Vilard de Lans, near Grenoble, France: Springer-Verlag, 1994.

[11] Curtis, B. *Superior Software Organizations*. in *Software Process Modelling in Practice*. Kensington Town Hall Conference Centre, London, UK: Butterworth-Heineman, 1993.

[12] Lehman, M.M. *The Role of Process Models in Software and Systems Development and Evolution*. in *Proceedings of the 5th International Software Process Workshop*. Kennebunkport, Maine, USA: IEEE Computer Society Press, 1989.

[13] Richardson, D.J., S.L. Aha, and L.J. Osterweil, *Integrating testing techniques through process programming. ACM SIGSOFT Software Engineering Notes*, Dec., 1989.

[14] Dyer, M., *et al.*, *Integrating an enterprise's engineering processes. Information and Software Technology: Special Issue Software Process Modelling in Practice*, **35**, 6/7, pp355-363, 1993.

[15] Serbanati, L.D., *Integrating Tools for Software Development.* Yourdon Press Computing Series, ed. E. Yourdon. Prentice-Hall: London, 1993.

[16] Lott, C.M. and H.D. Rombach, *Measurement-based guidance of software projects using explicit project plans. Information and Software Technology: Special Issue Software Process Modelling in Practice*, **35**, 6/7, pp407-419, 1993.

[17] Tate, G., G.M. Verner, and D.R. Jeffrey, *CASE: A Testbed for Modeling, Measurement and Management. Communications of the ACM*, **35**, 4, pp65-72, 1992.

[18] Tate, G., *Software process modelling and metrics: a CASE study. Information and Software Technology: Special Issue Software Process Modelling in Practice*, **35**, 6/7, pp323-330, 1993.

[19] Tully, C.J., ed. *Representing and Enacting the Software Process: Proceedings of the 4th International Software Process Workshop*, IEEE Computer Society Press: Moretonhampstead, Devon, England, 1988.

[20] Dowson, M., ed. *Proceedings of the First International Conference on the Software Process: Manufacturing Complex Systems*. IEEE Computer Society Press: 1991.

[21] Perry, D.E., ed. *Experience with Software Process Models: Proceedings of the 5th International Software Process Workshop*. IEEE Computer Society Press: Kennebunkport, Maine, USA, 1989.

[22] Potts, C., ed. *Proceedings of the First International Software Process Workshop*. IEEE Computer Society Press: Egham, Surrey, England, 1984.

[23] Dowson, M. and J.C. Wileden, *A Brief Report on the International Workshop on the Software Process and Software Environments. ACM SIGSOFT Software Engineering Notes*, **10**, 3, pp19-23, 1985.

[24] Dowson, M., ed. *Iteration in the Software Process: Proceedings of the 3rd International Software Process Workshop*. IEEE Computer Society Press: 1986.

[25] Schåfer, W. *European Process Work - Report from the 1st European Process Modeling Workshop*. in *Proceedings of the First International Conference on the Software Process: Manufacturing Complex Systems*. Redondo Beach, California, USA: IEEE Computer Society Press, 1991.

[26] Tully, C.J., ed. *Proceedings of the 4th International Software Process Workshop*. Moretonhampstead, Devon, England, 1988.

[27] Katayama, T., ed. *Proceedings of the 6th International Software Process Workshop: Support for the Software Process, held at Hokodate, Hokkaido, Japan, October 28-31 1990*. IEEE: 1991.

[28] Thomas, M.I. *The 7th International Software Process Workshop: Position Statement.* in *Proceedings of the First International Conference on the Software Process: Manufacturing Complex Systems*. Redondo Beach, California, USA: IEEE Computer Society Press, 1991.

[29] Warboys, B., ed. *Software Process Technology: Proceedings of the Third European Workshop EWSPT'94*. Lecture Notes in Computer Science, ed. G. Goos and J. Hartmanis. Vol. 772, Springer-Verlag: Berlin, 1994.

[30] Scherr, A.L., *A new approach to business processes. IBM Systems Journal*, **32**, 1, pp80-98, 1993.

[31] Mill, J., *No Pain, No Gain. in Computing:* pp26-27, 3 February, 1994.

[32] White, P., *Process Modelling and Process Support. IOPENER*, **1**, 3, pp7-8, 1992.

[33] Roberts, C. *Modelling and Coordinating Change in Business Processes*. in *Process Modelling Workshop for the BCS Bristol Branch*. Bristol University: Co-Ordination Systems Limited, 1993.

[34] Ince, D., *Tools of the Trade. in Computing:* pp28-29, 17 February, 1994.

[35] Ould, M. *An introduction to Process Modelling using RADs, illustrated by reference to the case study*. in *IOPTClub Practical Process Modelling*. Mountbatten Hotel, Monmouth Street, Covent Garden, London: 1992.

[36] Huckvale, T., M. McHugh, and C. Roberts. *Process Modelling Workshop*. in *Process Modelling Workshop for the BCS Bristol Branch*. Bristol University: Praxis Systems plc, 1993.

[37] Peltu, M., *Process Servers. in Computing:* pp36, 7 October, 1993.

[38] Classe, A., *CASE for Change. in Computing:* pp16-17, 5 August, 1993.

[39] Pocock, J.N. *VSF and its Relationship to Open Systems and Standard Repositories*. in *Software Development Environments and CASE Technology*. Konigswinter, Germany: Springer-Verlag, 1991.

[40] Kawalek, P. and P. White, *Supporting the business process. Computer Bulletin*, February, pp9-11, 1993.

[41] Fernstrom, C. *PROCESS WEAVER: Adding Process Support to Unix*. in *Proceedings of the Second International Conference on the Software Process*. Berlin, Germany: 1993.

[42] Fernstrom, C. *Process Centred Environments*. in *Software Process Modelling in Practice*. Kensington Town Hall Conference Centre, London, UK: Butterworth-Heineman, 1993.

[43] Henderson, P. *Software Processes are Business Processes Too*. in *Third International Conference on the Software Process*. Reston, Virginia, USA: IEEE Computer Society Press, 1994.

[44] Hammer, M. and J. Champy, *Reengineering the Corporation: A manifesto for change*. Nicholas Brealey publishing, London (by arrangement with Harper-Collins Publishers Inc., USA), 1993.

[45] Miers, D., *Use of Tools and Technology within a BPR Initiative,* in *Business Process Re-engineering: myth and reality,* Editor(s), C. Coulson-Thomas, Kogan Page: 1994.

[46] Massey, J., *Work to Rule. in Computing:* pp14-15, 2 September, 1993.

[47] Dowson, M. *Iteration in the Software Process: Workshop Introduction and Overview.* in *Proceedings of the 3rd International Software Process Workshop*. Breckenridge, Colorado, USA: IEEE Computer Society Press, 1986.

[48] Kellner, M.I. *Panel Session Position Paper: Software Process Modeling Experience.* in *Proceedings of the 11th International Conference on Software Engineering*. Pittsburgh, Pennsylvania: IEEE Computer Society Press, 1989.

[49] Kellner, M.I. *Software Process Modelling Support for Management Planning and Control.* in *Proceedings of the First IEEE International Conference on the Software Process*. Redondo Beach, California, USA: IEEE Computer Society Press, 1991.

[50] Kellner, M.I., *Representation Formalisms for Software Process Modelling. ACM SIGSOFT*, **14**, 4, pp93-96, 1988.

[51] Rombach, H.D. *Measurement-Based Guidance Using Explicit Project Plans.* in *Software Process Modelling in Practice*. Kensington Town Hall Conference Centre, London, UK: Butterworth-Heineman, 1993.

[52] Shepperd, M., *Products, processes and metrics. Information and Software Technology*, **34**, 10, pp674-680, 1992.

[53] Shepperd, M.J. *Quantitative approaches to process modelling.* in *Colloquium on Process Planning and Modelling*. London: IEE, 1992.

[54] Humphrey, W.S. *Review of State-of-the-Art: Session Summary.* in *Proceedings of the 5th International Software Process Workshop*. Kennebunkport, Maine, USA: IEEE Computer Society Press, 1989.

[55] Chandrashekar, S., B. Mayfield, and M. Samadzadeh, *Towards automating software project management. International Journal of Project Management*, **11**, 1, pp29-38, 1993.

[56] Humphrey, W.S. *The Software Engineering Process: Definition and Scope.* in *Proceedings of the 4th International Software Process Workshop*. Moretonhampstead, Devon, England: IEEE Computer Society Press, 1988.

[57] Basili, V.R. and H.D. Rombach, *Support for Comprehensive Reuse. Software Engineering Journal*, Sept., 1991.

[58] Redwine, S.T. *Software Reuse Processes.* in *Proceedings of the 4th International Software Process Workshop*. Moretonhampstead, Devon, England, UK: ACM, 1988.

[59] DTI, *TickIT Guide to Software Quality Management System and Certification using EN29001*. Department of Trade and Industry: 1990.

[60] Paulk, M.C., *Comparing ISO 9001 and the Capability Maturity Model for Software. Software Quality Journal*, **2**, pp245-256, 1993.

[61] Coallier, F., *How ISO 9001 fits into the Software World. IEEE Software*, Jan., pp98-100, 1994.

[62] Ince, D.C., *et al., ISO9000-3 and software measurement. BT Technology Journal*, **12**, 1, 1994.

[63] Bennington, H.D. *Production of large computer programs.* in *Proceedings ONR Symposium on Advanced Programming Methods for Digital Computers.* 1956.

[64] Royce, W.W. *Managing the development of large software systems.* in *Proceedings IEEE Wescon.* 1970.

[65] Sommerville, I., *Software Engineering: Fourth Edition*. International Computer Science Series, ed. A.D. McGettrick and J. van Leeuwen. Addison-Wesley: Wokingham, 1992.

[66] van Vliet, H., *Software Engineering: Principles and Practice*. John Wiley and Sons: Chichester, 1993.

[67] McCracken, D.D. and M.A. Jackson, *Life-cycle Concept Considered Harmful. ACM SIGSOFT Software Engineering Notes*, 7, 2, pp29-32, 1982.

[68] Gladden, G.R., *Stop the life-cycle, I want to get off. ACM SIGSOFT Software Engineering Notes*, 7, 2, 1982.

[69] Agresti, W.W., *The Conventional Software Life-cycle: Its evolution and assumptions,* in *New Paradigms for Software Development,* Editor(s), W.W. Agresti, IEEE Computer Society Press: 1986.

[70] Boehm, B.W., *et al., Characteristics of Software Quality*. North Holland: 1978.

[71] Boehm, B. and F.C. Belz. *Applying Process Programming to the Spiral Model.* in *Proceedings of the 4th International Software Process Workshop*. Moretonhampstead, Devon, England: IEEE Computer Society Press, 1988.

[72] Boehm, B.W., *A Spiral Model of Software Development and Maintenance. IEEE Computer*, **21**, 5, pp61-72, 1988.

[73] Boehm, B.W. and F. Belz. *Experiences with the Spiral Model as a Process Model Generator.* in *Proceedings of the 5th International Software Process Workshop*. Kennebunkport, Maine, USA: IEEE Computer Society Press, 1989.

[74] Madhavji, N.H., *The Process Cycle. Software Engineering Journal*, Sep, pp234-242, 1991.

[75] Osterweil, L.J. *Software Processes are Software Too.* in *Proceedings of the 3rd International Software Process Workshop*. Breckenridge, Colorado: IEEE Computer Society Press, 1986.

[76] Osterweil, L.J. *Experiences with Process Programming.* in *Proceedings of the 5th International Software Process Workshop*. Kennebunkport, Maine, USA: IEEE Computer Society Press, 1989.

[77] Heimbigner, D. *A logic language for process programming*. in *Proceedings of the 5th International Software Process Workshop*. Maine, USA: 1989.

[78] Heimbigner, D. *P^4: A Logic Language for Process Programming*. in *Proceedings of the 5th International Software Process Workshop*. Maine, USA: 1989.

[79] Heimbigner, D. *An Example P^4 Process Program for Rebus*. in *Proceedings of the 5th International Software Process Workshop*. Kennebunkport, Maine, USA: IEEE Computer Society Press, 1989.

[80] MacClean, R. *A Functional Paradigm for Software Development*. in *Proceedings of the 4th International Software Process Workshop*. Moretonhampstead, Devon, England, UK: ACM Press, 1988.

[81] Huseth, S. and D. Vines. *Describing the Software Process*. in *Proceedings of the 3rd International Software Process Workshop*. Breckenridge, Colorado, USA: IEEE Computer Society Press, 1986.

[82] Katayama, T. and M. Suzuki. *Mechanisms for Software Process Dynamics*. in *Proceedings of the 5th International Software Process Workshop*. Kennebunkport, Maine, USA: IEEE Computer Society Press, 1989.

[83] Katayama, T. *A hierarchical and functional software process description and its enaction*. in *Proceedings of the IEEE 11th International Conference on Software Engineering*. IEEE, 1988.

[84] Katayama, T. *A Hierarchical and Functional Approach to Software Process Description*. in *Proceedings of the 4th International Software Process Workshop*. Moretonhampstead, Devon, England: 1988.

[85] Katayama, T. and M. Suzuki. *An Example of Process Description in HFSP*. in *Proceedings of the 5th International Software Process Workshop*. Kennebunkport, Maine, USA: IEEE Computer Society Press, 1989.

[86] Kaiser, G.E. *Constructing Enactable Models*. in *Proceedings of the 4th International Software Process Workshop*. Moretonhampstead, Devon, England: IEEE Computer Society Press, 1988.

[87] Cheatham, T.E. *Activity Coordination Programs*. in *Proceedings 4th International Software Process Workshop*. Moretonhampstead, Devon, England: 1988.

[88] Conradi, R. *Methodology Session*. in *Third European Workshop on Software Process Technology EWSPT'94*. Villard de Lans, France: Springer-Verlag, 1994.

[89] Conradi, R. *Meta-Process / Methodology Session*. in *Proceedings of the Third European Workshop on Software Process Technology*. Vilard de Lans, near Grenoble, France: Springer-Verlag, 1994.

[90] Conradi, R., C. Fernstrom, and A. Fuggetta, *A Conceptual Framework for Evolving Software Processes. ACM SIGSOFT Software Engineering Notes*, **18**, 4, pp26-35, 1993.

[91] Osterweil, L.J. *Example Process Program Code, Coded in Appl/A*. in *Proceedings of the 5th International Software Process Workshop*. Kennebunkport, Maine, USA: IEEE Computer Society Press, 1989.

[92] Ohki, A. and K. Ochimizu. *Process Programming with Prolog*. in *Proceedings of the 4th International Software Process Workshop*. Moretonhampstead, Devon, England: ACM Press, 1988.

[93] Derniame, J.C. *Process Modelling Issues Session*. in *Third European Workshop on Software Process Technology EWSPT'94*. Villard de Lans, France: Springer-Verlag, 1994.

[94] Engels, G. et al. *On the Structure of an Incremental and Integrated Software Development Environment,* Proceedings of the 19th Annual Hawaii Conference on System Sciences, Hawaii, USA, 1986.

[95] Dowson, M. *ISTAR - An Integrated Project Support Environment*. in *Proceedings of the ACM SIGSOFT/SIGPLAN Symposium on Practical Software Development Environments*. 1986.

[96] Junkermann, G. and W. Schafer. *A Design Methodology for Process-Programming*. in *Proceedings of the Third European Workshop on Software Process Technology*. Vilard de Lans, near Grenoble, France: Springer-Verlag, 1994.

[97] Balzer, R. *Process Programming: Passing into a New Phase*. in *Proceedings of the 4th International Software Process Workshop*. Moretonhampstead, Devon, England: IEEE Computer Society Press, 1988.

[98] Lehman, M.M. *Models in Software Development and Evolution*. in *Software Process Modelling in Practice*. Kensington Town Hall Conference Centre, London, UK: Butterworth-Heineman, 1993.

[99] Lehman, M.M. *Some Reservations on Software Process Programming*. in *Proceedings of the 4th International Software Process Workshop*. Moretonhampstead, Devon, England, UK: ACM Press, 1988.

[100] DTI, *The STARTS Guide*. DTI: 1987 (now published by NCC).

[101] Ashok, V., *et al. Process Modelling in Software Environments*. in *Proceedings of the 4th International Software Process Workshop*. Moretonhampstead, Devon, England: IEEE Computer Society Press, 1988.

[102] Huff, K.E. *Probing Limits to Automation: Towards Deeper Process Models*. in *Proceedings of the 4th International Software Process Workshop*. Moretonhampstead, Devon, England: ACM Press, 1988.

[103] Huff, K.E. *Software Process Instantiation and the Planning Paradigm*. in *Proceedings of the 5th International Software Process Workshop*. Kennebunkport, Maine, USA: IEEE Computer Society Press, 1989.

[104] Huff, K.E. *GRAPPLE Example: Processes as Plans*. in *Proceedings of the 5th International Software Process Workshop*. Kennebunkport, Maine, USA: IEEE Computer Society Press, 1989.

[105] Hitchcock, P. *The Process Model of the Aspect IPSE*. in *Proceedings of the 4th International Software Process Workshop*. Moretonhampstead, Devon, England: IEEE Computer Society Press, 1988.

[106] Chroust, G., *Application Development Project (ADPS) Support an Environment for Industrial Application Development. ACM SIGSOFT Software Engineering Notes*, July, 1989.

[107] McDermid, J. and K. Ripken, *Life-cycle Support in the Ada Environment*. Cambridge University Press: 1984.

[108] Dell, P.W., *Early Experience with an IPSE. Software Engineering Journal*, pp259-264, 1986.

[109] Bott, M.F., *ECLIPSE - An Integrated Project Support Environment*. Peter Peregrinus: 1989.

[110] Higgs, M. and P. Rook, *A Generalized IPSE Architecture derived from the principles of Software Engineering and its implementation in Genos*. GEC Software Ltd, 1987.

[111] Aldeson, A., M.F. Bott, and M.E. Falla, *The Eclipse Object Management System. Software Engineering Journal*, Jan., 1986.

[112] Shen, V., *What happened to development environments. IEEE Software*, pp120 & 124, 1991.

[113] Notkin, D. *Applying Process Models to the Full Lifecycle is Premature*. in *Proceedings of the 4th International Software Process Workshop*. Moretonhampstead, Devon, England: ACM Press, 1988.

[114] Riddle, W.E. *Existing Approaches: Session Summary*. in *Proceedings of the 3rd International Software Process Workshop*. Breckenridge, Colorado, USA: IEEE Computer Society Press, 1986.

[115] Williams, L.G. *Software Process Modelling: A Behavioural Approach*. in *Proceedings of the 10th International Conference on Software Engineering*. Singapore: IEEE, 1988.

[116] Schafer, W. *ESF and software process modeling*. in *Proc. 5th International Software Process Workshop*. 1989.

[117] Warboys, B., *The IPSE 2.5 Project: Process Modelling as the basis for a Support Environment*, in *Proceedings of the First International Conference on System Development Environments and Factories*, Editor(s), N. Madhavji, W. Schafer, and H. Weber, Pitman: London, 1989.

[118] Bandinelli, S., A. Fuggetta, and C. Ghezzi, *Software Process Model Evolution in the SPADE Environment. IEEE Transactions on Software Engineering*, **19**, 12, pp1128-1144, 1993.

[119] Kaiser, G.E. and P.H. Feiler. *An Architecture for Intelligent Assistance in Software Development*. in *Proceedings of the 9th International Conference on Software Engineering*. Monterey, California, USA: IEEE, 1987.

[120] Kaiser, G.E. *Rule-Based Modelling of the Software Development Process*. in *Proceedings of the 4th International Software Process Workshop*. Moretonhampstead, Devon, England: 1988.

[121] Kaiser, G.E., P.H. Feiler, and S.S. Popovich, *Intelligent Assistance for Software Development and Maintenance. IEEE Software*, May, pp40-49, 1988.

[122] Kaiser, G. *Experience with Marvel.* in *Proceedings of the 5th International Software Process Workshop*. Kennebunkport, Maine, USA: IEEE Computer Society Press, 1989.

[123] Kaiser, G.E. *Marvel Strategy Language Example.* in *Proceedings of the 5th International Software Process Workshop*. Kennebunkport, Maine, USA: IEEE Computer Society Press, 1989.

[124] Kaiser, G.E., N.S. Barghouti, and M.H. Sokolsky. *Preliminary experience modeling in the Marvel software development environment.* in *Proc. 23rd IEEE Annual Hawaii International Conference on System Sciences*. Hawaii: 1990.

[125] Deiters, W. and V. Gruhn, *Managing software processes in the environment MELMAC. ACM SIGSOFT Software Eng. Notes*, Dec., pp193-205, 1990.

[126] Bandinelli, S., *et al. The Architecture of the SPADE-1 Process Centred SEE.* in *Proceedings of the Third European Workshop on Software Process Technology*. Vilard de Lans, near Grenoble, France: Springer-Verlag, 1994.

[127] Estublier, J. *Architecture Session.* in *Third European Workshop on Software Process Technology EWSPT'94*. Villard de Lans, France: Springer-Verlag, 1994.

[128] Robertson, I. *An Implementation of the ISPW-6 Process Example.* in *Proceedings of the Third European Workshop on Software Process Technology.* Vilard de Lans, near Grenoble, France: Springer-Verlag, 1994.

[129] Snowdon, R.A., *A Brief Overview of the IPSE 2.5 Project. Ada User*, **9**, 4, pp156-161, 1988.

[130] Schåfer, W., *et al. ESF and Software Process Modeling.* in *Proceedings of the 5th International Software Process Workshop*. Kennebunkport, Maine, USA: IEEE Computer Society Press, 1989.

[131] Lonchamp, J., *et al., Towards assisted software engineering environments. Information & Software Technology*, **33**, 8, pp581-593, 1991.

[132] Gillies, A.C., *Humanization of the software factory. Information and Software Technology*, **33**, 9, pp641-646, 1991.

[133] Levendel, Y., *Improving Quality with a Manufacturing Process. IEEE Software*, March, pp13-25, 1991.

[134] Fernstrom, C., K. Narfelt, and L. Ohlsson, *Software Factory Principles, Architecture, and Experiments. IEEE Software*, Mar, pp36-44, 1992.

[135] Fernstrom, C., *Tools and environments to improve the software process. Information and Software Technology*, **34**, 10, pp659-673, 1992.

[136] Ould, M.A. and C. Roberts. *Modelling Iteration in the Software Process.* in *Proceedings of the 3rd International Software Process Workshop*. Breckenridge, Colorado, USA: IEEE Computer Society Press, 1986.

[137] Roberts, C. *Describing and Acting Process Models with PML.* in *Proceedings of the 4th International Software Process Workshop.* Moretonhampstead, Devon, England: 1988.

[138] Roberts, C. and A. Jones. *Dynamics of Process Models in PML.* in *Proceedings of the 5th International Software Process Workshop.* Kennebunkport, Maine, USA: IEEE Computer Society Press, 1989.

[139] Snowdon, R.A. and C. Roberts. *Dynamics of Process Models in PML.* in *Proceedings of the 5th International Software Process Workshop.* Kennebunkport, Maine, USA: IEEE Computer Society Press, 1989.

[140] Warboys, B. *Open Issues Session.* in *Third European Workshop on Software Process Technology EWSPT'94.* Villard de Lans, France: Springer-Verlag, 1994.

[141] Dorling, A. *SPICE (Software Process and Capability dEtermination).* in *Software Process Modelling in Practice.* Kensington Town Hall Conference Centre, London, UK: Butterworth-Heineman, 1993.

[142] Dorling, A., *SPICE: Software Process Improvement and Capability dEtermination. Software Quality Journal*, **2**, pp209-224, 1993.

[143] Francis, R., *Quality and process management: a view from the UK computing services industry. Software Quality Journal*, **2**, pp225-238, 1993.

[144] Koch, G. *Process Assessment: The 'BOOTSTRAP' Approach.* in *Software Process Modelling in Practice.* Kensington Town Hall Conference Centre, London, UK: Butterworth-Heineman, 1993.

[145] Bell(Canada), *TRILLIUM-Telecom software product development capability assessment model, Draft 2.1.* Technical Report, Bell Canada, 1992.

[146] Craigmyle, M. and I. Fletcher, *Improving IT effectiveness through software process assessment. Software Quality Journal*, **2**, pp257-264, 1993.

[147] Humphrey, W.S., *Characterising the software process: a maturity framework. IEEE Software*, **5**, 2, pp73-79, 1988.

[148] Humphrey, W.S., *Managing the Software Process.* Addison-Wesley: 1989.

[149] Paulk, M.C., B. Curtis, and M.B. Chrissis, *Capability Maturity Model for Software.* Technical Report CMU/SEI-91-TR-24, Software Engineering Institute, 1991.

[150] Radice, R.A., *et al., A programming process architecture. IBM Systems Journal*, **24**, 2, pp79-90, 1985.

[151] Crosby, P., *Quality is Free.* McGraw-Hill: 1979.

[152] Curtis, B. and M. Paulk, *Creating a software process improvement program. Information and Software Technology: Special Issue Software Process Modelling in Practice*, **35**, 6/7, pp381-386, 1993.

[153] Paulk, C.M., *et al., Capability maturity model for software Version 1.1.* Technical Report CMU/SEI-93-TR-24, Carnegie-Mellon University, Software Engineering Institute, 1993.

[154] Rowe, A. and R. Witty, *Ami: promoting a quantitative approach to software management. Software Quality Journal*, **2**, pp291-296, 1993.

[155] Kuntzmann-Combelles, A., *et al.*, ed. *Handbook of the Application of metrics in Industry: A quantitative approach to software management.* AMI ESPRIT project: 1992.

[156] Pfleeger, S.L. and C. McGowan, *Software metrics in the process maturity framework. J. of Systs. Software*, 12, pp255-261, 1990.

[157] Pfleeger, S.L., *Process maturity as a framework for CASE tool selection. Information and Software Technology*, **33**, 9, pp611-615, 1991.

[158] Card, D., *Understanding Process Improvement. IEEE Software*, **8**, 7, pp102-103, 1991.

[159] Thompson, K. *A method for assessing organizational software development capability.* in *Software Tools*. Institute of Software Engineering, Belfast, 1991.

[160] Thompson, K., *Practical Quality Improvement Through Software Process Maturity*. Technical Report, Institute of Software Engineering, Belfast, 1992.

[161] Thompson, K. *Software Process Maturity and the Information Systems Developer.* in *Software Process Modelling in Practice*. Kensington Town Hall Conference Centre, London, UK: Butterworth-Heineman, 1993.

[162] Thompson, K. and P. McParland, *Software Process Maturity (SPM) and the information systems developer. Information and Software Technology: Special Issue Software Process Modelling in Practice*, **35**, 6/7, pp331-338, 1993.

[163] Bollinger, T.B. and C. McGowan, *A Critical Look at Software Capability Evaluations. IEEE Software*, July, pp25-41, 1991.

[164] Finklestein, A., *A Software Process Immaturity Model. ACM SIGSOFT Software Engineering Notes*, Oct., pp22-23, 1992.

[165] Baumert, J., *New SEI Maturity Model Targets Key Practices. IEEE Software*, Nov., pp78-9, 1991.

[166] Dion, R., *Elements of a Process-Improvement Program. IEEE Software*, July, pp83-85, 1992.

[167] Gilchrist, J.M., *Project evaluation using the SEI method. Software Quality Journal*, **1**, pp37-44, 1992.

[168] Snowdon, P. and B. Grover, *How Texaco is Getting to Level 3 of SPMM. Software Development*, June, pp181-186, 1992.

[169] Kitson, D.H. and S.M. Masters. *An Analysis of SEI Software Process Assessment Results: 1987-1991.* in *15th International Conference on Software Engineering*. Baltimore, Maryland: IEEE Computer Society Press, 1993.

[170] Lott, C.M. and H.D. Rombach. *Measurement-Based Guidance Through Explicit Process Models.* in *Software Process Modelling in Practice.*

Kensington Town Hall Conference Centre, London, UK: Butterworth-Heineman, 1993.

[171] Krasner, H., *et al.*, *Lessons Learned from a Software Process Modelling System. Communications of the ACM*, **35**, 9, pp91, 1992.

[172] Sommerville, I. *Experience Session.* in *Third European Workshop on Software Process Technology EWSPT'94*. Villard de Lans, France: Springer-Verlag, 1994.

[173] Ambriola, V. *Applying a Metric Framework to the Software Process: An Experiment.* in *Proceedings of the Third European Workshop on Software Process Technology*. Vilard de Lans, near Grenoble, France: Springer-Verlag, 1994.

[174] Basili, V.R. and H.D. Rombach. *Tailoring the Software Process to Project Goals and Environments.* in *Proceedings 9th International Conference on Software Engineering*. Monterey: 1987.

[175] Basili, V.R. and H.D. Rombach, *The TAME Project: Towards Improvement-Oriented Software Environments. IEEE Trans. on Software Engineering*, **14**, 6, pp758-773, 1988.

[176] Phalp, K.T., *An Investigation into Potential Metric Usage at a Sclumberger Software Project.* Bournemouth Polytechnic, 1991.

[177] Thomas, I. *The Software Process as a Goal-Directed Activity.* in *Proceedings of the 5th International Software Process Workshop*. Kennebunkport, Maine, USA: IEEE Computer Society Press, 1989.

[178] Arbaoui, S. and F. Oquenedo. *Goal Oriented vs Activity Oriented Process Modelling and Enactment: Issues and Perspectives.* in *Proceedings of the Third European Workshop on Software Process Technology.* Vilard de Lans, near Grenoble, France: Springer-Verlag, 1994.

[179] Walrad, C. and E. Moss, *Measurement: The key to application development quality. IBM Systems Journal*, **32**, 3, pp445-460, 1993.

[180] Pfleeger, S.L., J.C.J. Fitzgerald, and D.A. Rippy, *Using multiple metrics for analysis of improvement. Software Quality Journal*, **1**, pp27-36, 1992.

[181] Pfleeger, S.L., *Lessons Learned in Building a Corporate Metrics Program. IEEE Software*, May, pp67-74, 1993.

[182] Mi, P. and W. Scacchi. *Modeling Articulation Work in Software Engineering Processes.* in *Proceedings of the First IEEE International Conference on the Software Process*. Redondo Beach, California, USA: IEEE Computer Society Press, 1991.

[183] Curtis, B. *Models of Iteration in Software Development.* in *Proceeding of the 3rd International Software Process Workshop*. Breckenridge, Colorado, USA: IEEE Computer Society Press, 1986.

[184] Curtis, B., M.I. Kellner, and J. Over, *Process Modelling*. *Communications of the ACM*, **35**, 9, pp75-90, 1992.

[185] Devaranne, A. and C. Ozanne. *The Need of a Process Engineering Method.* in *Proceedings of the Second European Workshop on Software Process Technology, EWSPT'92*. Trondeheim, Norway: Springer-Verlag, 1992.

[186] Conradi, R., *et al. Towards a Reference Framework for Process Concepts.* in *Proceedings of the European Workshop on Software Process Technology*. Trondeheim, Norway, September 92: Springer-Verlag, 1992.

[187] Longchamp, J. *A Structured Conceptual and Terminoligical Framework for Software Process Engineering.* in *Proceedings of the Second International Conference on the Software Process*. Berlin: IEEE Computer Society Press, 1993.

[188] Jaccheri, M.L. and R. Conradi, *Techniques for Process Model Evolution in EPOS. IEEE Transactions on Software Engineering*, **19**, 12, pp1145-1156, 1993.

[189] Kawalek, P.J. *The Process Modelling Cookbook Orientation, Description and Experience.* in *Software Process Technology: Second European Workshop, EWSPT '92*. Trondheim, Norway: 1992.

[190] Kawalek, P. and I. Robertson, *Case Study Contributions to Process Modelling Methodology. IOPENER*, **1**, 5, pp8-9, 1992.

[191] Sommerville, I. and T. Rodden. *Understanding the Software Process as a Social Process.* in *Proceedings of the Second European Workshop on Software Process Technology, EWSPT'92*. Trondeheim, Norway: Springer-Verlag, 1992.

[192] Curtis, B. *Building software processes under the lamppost.* in *Proceedings of the Ninth International Conference on Software Engineering*. Monterey, CA: 1987.

[193] Curtis, B., D. Walz, and J. Elam. *Studying the Process of Software Design Teams.* in *Proceedings of the 5th International Software Process Workshop*. Kennebunkport, Maine, USA: IEEE Computer Society Press, 1989.

[194] Curtis, B., *Empirical studies of the software design process,* in *Proc. INTERACT '90,* Editor(s), D. Diaper, Elsevier Science: 1990.

[195] Basili, V.R. and D. Hutchens, *An empirical study of a syntactic complexity family. IEEE Transactions on Software Engineering*, **9**, 6, pp639-647, 1983.

[196] Curtis, *Video Address to the Psychology of Programming Group at Warwick University*, 1989.

[197] McGowan, C. and L. Bohner. *Model-based Process Improvement.* in *Proceedings of the 15th International Conference on Software Engineering*. Baltimore, Maryland, USA: IEEE Computer Society Press, 1993.

[198] Warboys, B. *Wrap up Session.* in *Third European Workshop on Software Process Technology EWSPT'94*. Villard de Lans, France: Springer-Verlag, 1994.

[199] Bradac, M., D.E. Perry, and L.G. Votta. *Prototyping a Process Monitoring Experiment.* in *Proceedings of the 15th International Conference on*

Software Engineering. Baltimore, Maryland, USA: IEEE Computer Society Press, 1993.

[200] Perry, D.E., N.A. Staudenmayer, and L.G. Votta, *People, Organizations, and Process Improvement. IEEE Software*, July, 1994.

[201] Perry, D.E., Humans in the Process: Architectural Implications, Proceedings of the 8th International Software Workshop, Schloss Dagstuhl, Germany, March 1993.

[202] Yourdon, E., *Modern Structured Analysis*. Prentice-Hall International Editions, Prentice Hall: 1989.

[203] Furnas, G.W. *Generalized Fisheye Views*. in *Proceedings of the ACM SIGCHI'86 Conference on Human Factors in Computing Systems*. 1986.

[204] Churcher, N.I., *Photi-A Fisheye View of Bubbles*. Technical Report TR-COSC-15/93, Department of Computer Science, University of Canterbury, Christchurch, New Zealand, 1993.

[205] Sarkar, M. and M.H. Brown. *Graphical Fisheye Views of Graphs*. in *Proceedings of the ACM SIGCHI'92*. 1992.

[206] Mackinlay, J.D., G.G. Robertson, and S.K. Card. *The perspective wall: Detail and context smoothly integrated.* in *Proceedings of the ACM SIGCHI'91 Conference on Human Factors in Computing Systems*. 1991.

[207] Saeki, M., T. Kaneko, and M. Sakamoto. *A Method for Software Process Modeling and Description using LOTOS*. in *Proceedings of the First IEEE International Conference on the Software Process*. Redondo Beach, California, USA: IEEE Computer Society Press, 1991.

[208] Harel, D., *Biting the Silver Buller. IEEE Computer*, Jan., pp8-20, 1992.

[209] Harel, D., *On Visual Formalisms. Communications of the ACM*, **31**, 5, pp514-530, 1988.

[210] Kellner, M.I. *Experience with Enactable Software Process Models*. in *Proceedings of the 5th International Software Process Workshop*. Kennebunkport, Maine, USA: IEEE Computer Press, 1989.

[211] Kellner, M.I. and G.A. Hansen. *Software Process Modeling: A Case Study*. in *Proceedings of the 22nd IEEE Annual Hawaii International Conference on System Sciences*. Hawaii: IEEE, 1989.

[212] Harel, D., *et al.*, *STATEMATE A Working Environment for the Development of Complex Reactive Systems. IEEE Transactions on Software Engineering*, **16**, 10, pp403-414, 1988.

[213] Phillips, T., *Raving about Statemate. in Computing:* pp18, 27 May, 1993.

[214] Sugiyama, Y. and E. Horowitz. *OPM: An Object Process Modelling Environment*. in *Proceedings of the 5th International Software Process Workshop*. Kennebunkport, Maine, USA: IEEE Computer Society Press, 1989.

[215] Pengelly, A. *Software Process Modelling within BT*. in *Software Process Modelling in Practice*. Kensington Town Hall Conference Centre, London, UK: Butterworth-Heineman, 1993.

[216] Phalp, K. and M. Shepperd. *A Pragmatic Approach to Process Modelling*. in *Proceedings of the Third European Workshop on Software Process Technology*. Vilard de Lans, near Grenoble, France: Springer-Verlag, 1994.

[217] Starke, G. and M. von der Beeck. *SA/CM/IM for Process Modelling*. in *Proceedings of the Second European Workshop on Software Process Technology, EWSPT'92*. Trondheim, Norway: Springer-Verlag, 1992.

[218] Starke, G., *Urgent Research Issues in Software Process Engineering*. *Communications of the ACM*, **18**, 4, pp13-15, 1993.

[219] Pyzdek, T., *To Improve Your Process: Keep it Simple. IEEE Software*, Sep, pp112-113, 1992.

[220] Tamai, T., *Current practices in software processes for systems planning and requirements analysis. Information and Software Technology: Special Issue Software Process Modelling in Practice*, **35**, 6/7, pp339-344, 1993.

[221] Hammersly, M., *what's wrong with Ethnography*? Routledge: 1992.

[222] Smith, N.C., *The case study: a useful research method for information management. Journal of Information Technology*, **5**, pp123-133, 1990.

[223] Yin, R.K., *Case Study Research: Design and Methods*. 1984.

[224] Adelman, L., *Experiments, Quasi-Experiments and Case Studies: A Review of Empirical Methods for Evaluating Decision Support Systems. IEEE Transactions on Systems, Man and Cybernetics*, **21**, 2, pp293-301, 1991.

[225] Pfleeger, S.L., *Design and Analysis in Software Engineering. ACM SIGSOFT Software Engineering Notes*, **19**, 4, pp16-20, 1994.

[226] Worsley, P., *Introducing Sociology*. Penguin: 1970.

[227] Mitchell, J.C., *Case and situation analysis. The Sociological Review*, 31, pp187-211, 1983.

[228] DESMET, *Case Study Design and Analysis Procedures (CSDA)*. National Computing Centre Ltd (NCC): 1994.

[229] Sommerville, *Re: Field Studies, case studies in software engineering*. 23 December 94, 1994.

[230] Potts, C., *Software-Engineering Research Revisited. IEEE Software*, Sep, pp19-28, 1993.

[231] Co-Ordination, *RADitor version 1.5: Users Manual*. Co-Ordination Systems: 1994.

[232] ICL, *Process Wise Workbench: Version 5.2 Users Guide*. ICL: 1995.

[233] Hatley, D.J. and I.A. Pirbhai, *Strategies for Real-Time System Specification*. Dorset House Publishing: New York, 1987.

[234] Kramer, B. and Luqi. *Petri-Net Based Models of Software Engineering Processes*. in *Proceedings of the 23rd Annual Hawaii International Conference on System Sciences*. Hawaii: 1990.

[235] Coulson-Thomas, C., *Business Process Re-engineering: myth and reality*. Kogan Page: 1994.

[236] Hoare, C.A.R., *Communicating Sequential Processes*. Prentice-Hall International series in Computer Science, Prentice-Hall: 1985.

[237] Henderson, P., *The CSP Stepper in Enact - an Executable Specification: available as an ftp source from ftp.ecs.soton.ac.uk in pub/peter/various/csp.ps./*. 1992.

[238] Henderson, P., *Object-Oriented Specification and Design with C++*. The McGraw -Hill International Series in Software Engineering, McGraw-Hill: 1993.

[239] Schlaer, S. and S.J. Mellor, *An object-orientated approach to domain analysis*. *ACM SIGSOFT Software Engineering Notes*, **14**, 5, pp66-77, 1989.

[240] Fuggetta, A., *et al.*, *Executable Specifications with Data-flow Diagrams*. *Software Practice and Experience*, **23**, 6, pp629-652, 1993.

[241] Glass, R.L., *Pilot Studies: What, Why and How*. *The Software Practitioner*, Jan., 1995.

[242] Boehm, B.W., *Software Engineering Economics*. Prentice-Hall: Englewood Cliffs, N.J., 1981.

[243] Rombach, H.D. *Specification for Software Process Measurement*. in *Proceedings of the 5th International Software Process Workshop*. Kennebunkport, Maine, USA: IEEE Computer Society Press, 1989.

[244] Ould, M.A., *Business Processes: Modelling and Analysis for Re-engineering and Improvement*. Wiley: Chichester, 1995.

[245] Parnas, D.L. and P.C. Clements, *A Rational Design Process: How and Why to Fake It*. *IEEE Trans. on Softw. Eng.*, **12**, 2, pp251-257, 1986.

[246] Rifkin, S. and C. Cox, *Measurement in Practice*. Technical Report CMU/SEI-91-TR-16, Software Engineering Institute, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 1991.

[247] Davis, C.J., *et al.*, *Current practice in software quality and the impact of certification schemes*. *Software Quality Journal*, **1**, 2, pp145-161, 1991.

[248] Hunter, R.B. and A.K. Rae, *Software Quality Management*. *IEEE*, 1993.

[249] ACARD, *Software: A Vital Key to UK Competitiveness*, Advisory Council for Applied Research and Development, 1986.

[250] Logica, *Quality Management Standards for Software*, Logica Consultancy Limited, 1988.

[251] Price-Waterhouse, *Software Quality Standards: The Cost and Benefits*, Price Waterhouse, Management Consultants, 1988.

[252] Avison, D.E., H.U. Shah, and D.N. Wilson, *Software quality standards in practice: the limitations of using ISO-9001 to support software development. Software Quality Journal*, **1**, 3, pp105-111, 1994.

[253] Koch, G.R., *Process assessment: the 'BOOTSTRAP' approach. Information and Software Technology: Special Issue Software Process Modelling in Practice*, **35**, 6/7, pp387-403, 1993.

[254] Lehman, M.M., *Human Thought and Action as an Ingredient of System Behaviour,* in *The Encyclopaedia of Ignorance,* Editor(s), R. Duncan and M. Weston-Smith, Pergamon Press: Oxford, 1977.

[255] Lehman, M.M., *Software engineering, the software process and their support. Software Engineering Journal*, **6**, 5, pp243-258, 1991.

Appendices

The appendices contain documents that were produced at the time of the study.

These documents have not be altered or tidied up for the write-up of the thesis, and hence they constitute a source of evidence for the claims made by the thesis.

For example, we include the original 'Teamwork' diagrams produced for the exploratory study. Similarly we show the 5 project (instance) model at the same scale. We also include some documents which were produced for the organization, the data collected from projects and some of our own notes on the cases investigated.

Appendix A: Models Produced During the Exploratory Study

This Appendix contains the original data flow models produced for the exploratory study. These were produced using the CASE tool 'Teamwork', and, therefore, use a hierarchical numbering system. Thus, the children of activity 3, will be 3.1, 3.2, 3.3 etc.

The models reflect the five phases of the exploratory investigation. That is models based upon:

Theoretical or documented process

1) Procedures documents alone.

2) Procedures and templates

Actual process

3) Documents output by the process

4) Interviews

5) Validation interviews

DFD 0 . . . . . Procedures (8, 9, 10)
DFD 1 . . . . . Concept (6)
DFD 2 . . . . . Feasibility (11)
DFD 2.7 . . . . . Develop_Blue_Book (5)

ocedures
PHASE ONE

Product_Extension_Requirement

To be used for asking questions
and clarifying current process

White_Book

Feasability
2

Blue_Book

Design
3

Product_Definitions

Market_Need

No mention of
risk matrix

Concept
1

Product_Proposal_Acceptance

Launch Decision

(To whom?)

Product_Proposal
_Rejection

Customer_
Requirement

Product_Idea/concept

Project_
Control
8

Evaluation
9

Production_
and_PDS_
handover_doc

Verified_&_Accepted_
Production_Standard_Design

Modifications

Tested_Production
_Models

Support
7

User_Training_Preparations

Customer_Support_docs

Production_Process
_docs

Development
4

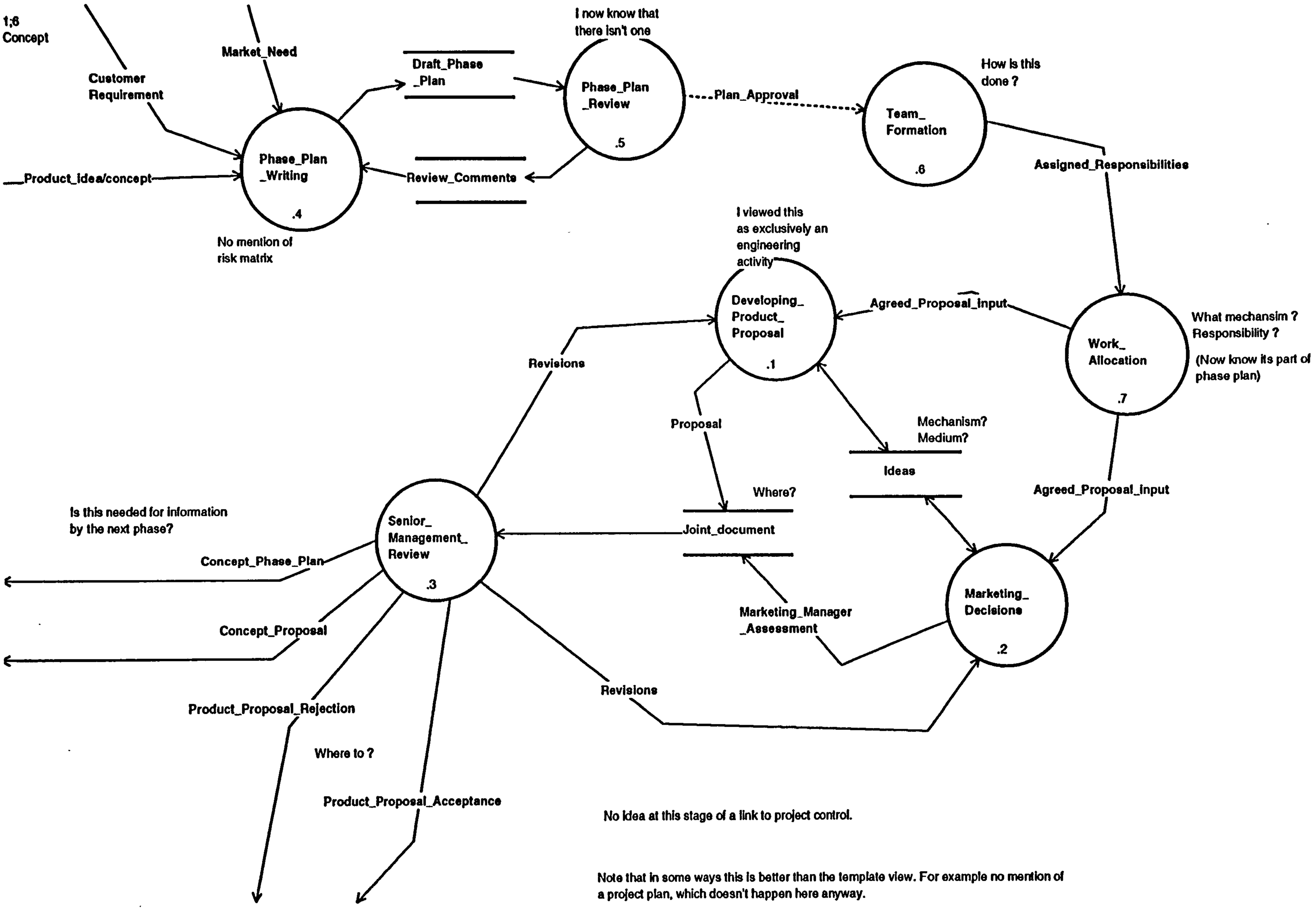
Tested_Components_
and_test_docs

Acceptance_Test_Results

Acceptance
6

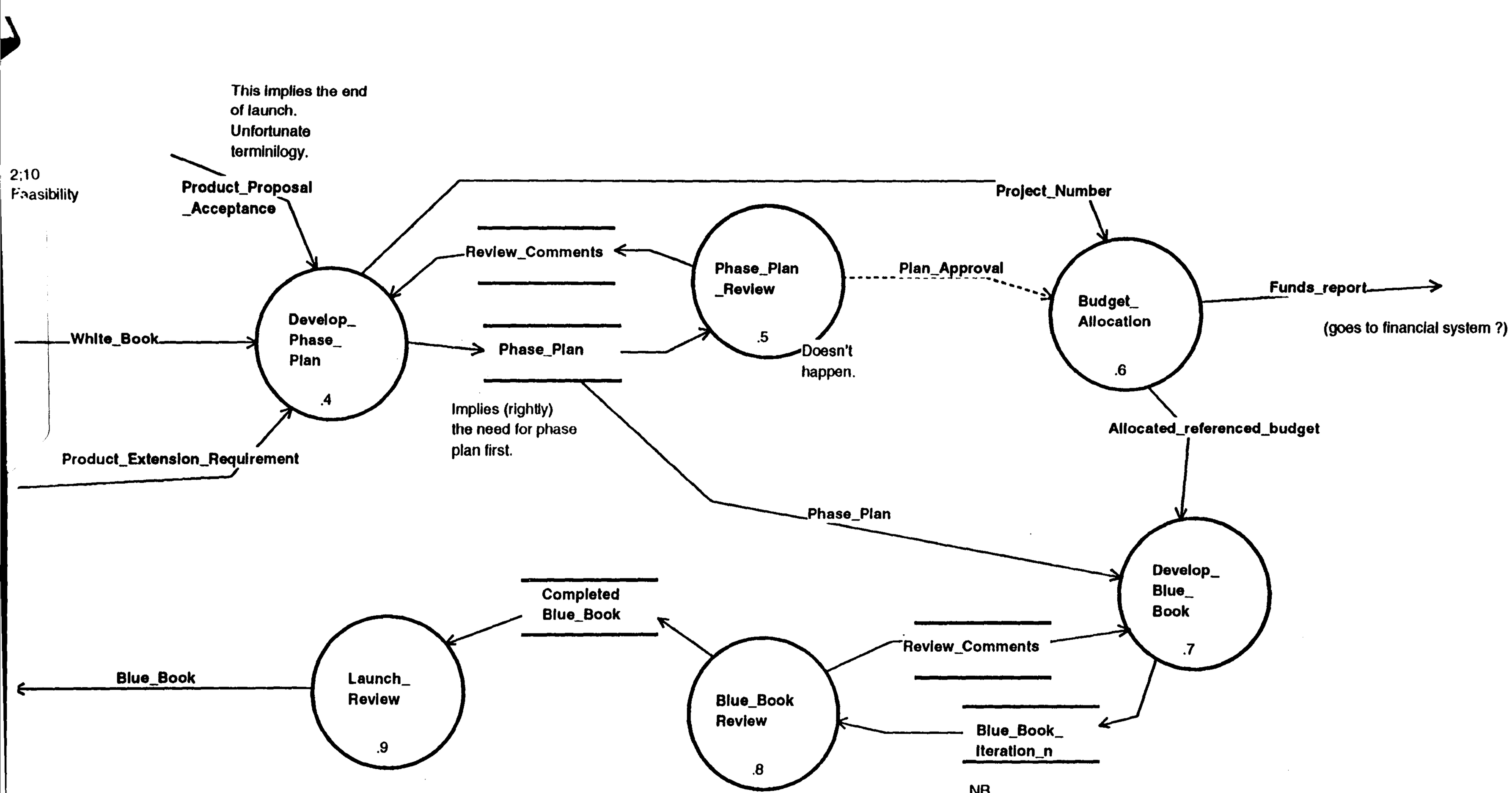
Production
5

Acceptance
_Test_Procedures



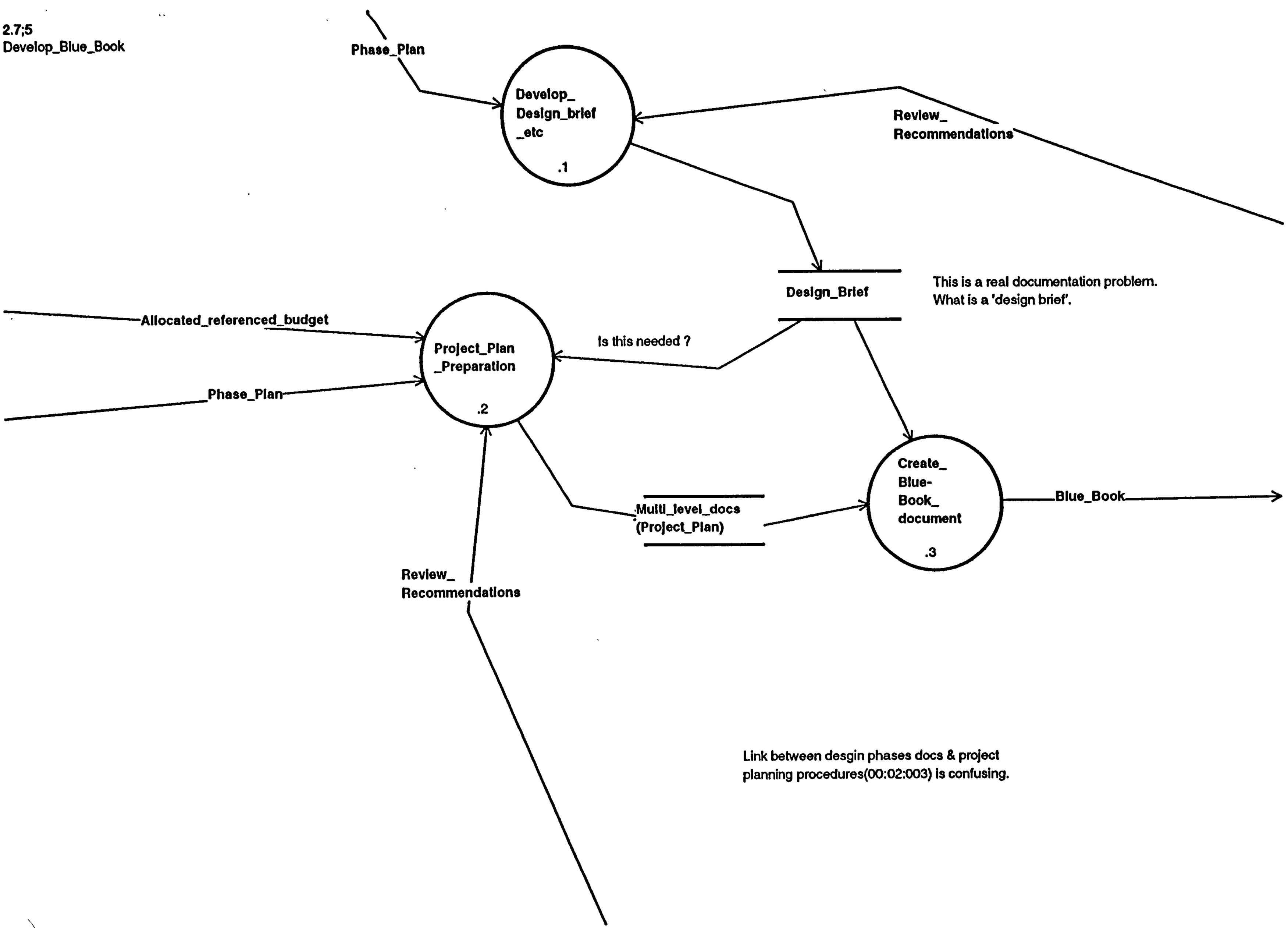
No idea at this stage of a link to project control.

Note that in some ways this is better than the template view. For example no mention of a project plan, which doesn't happen here anyway.



NB
This idea of iteration was one that almost became lost in looking at templates, and talking to users of the process.
However, its inclusion in this first model shows that it is an idea which I picked up from reading the procedures documents.

2.7;5
Develop_Blue_Book



Blue_Book (data flow) =

Feasability_Phase_Plan +

Product_Proposal +

Project_Plan

White_Book (data flow) =

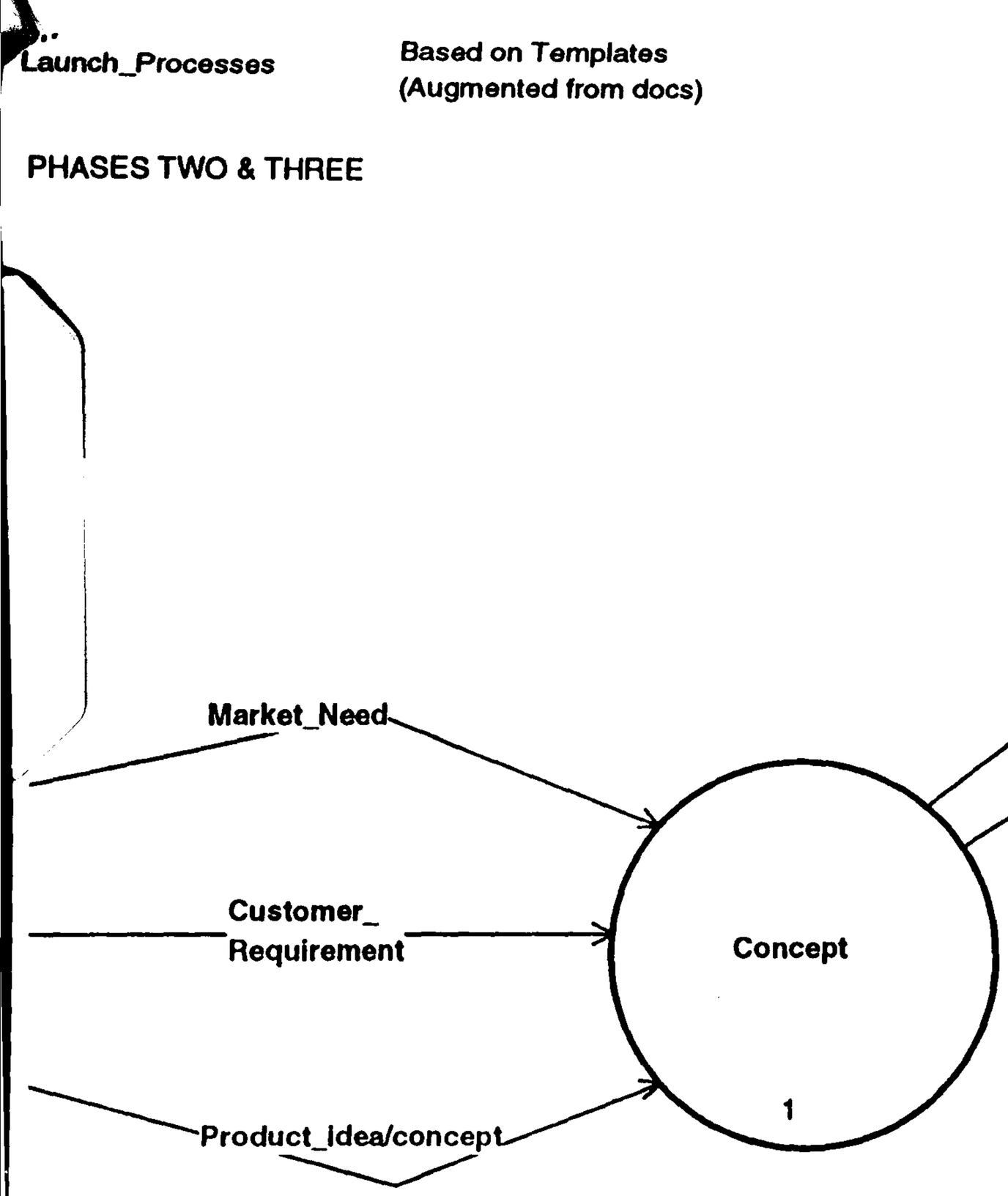
Concept_Proposal +
Phase_Plan +

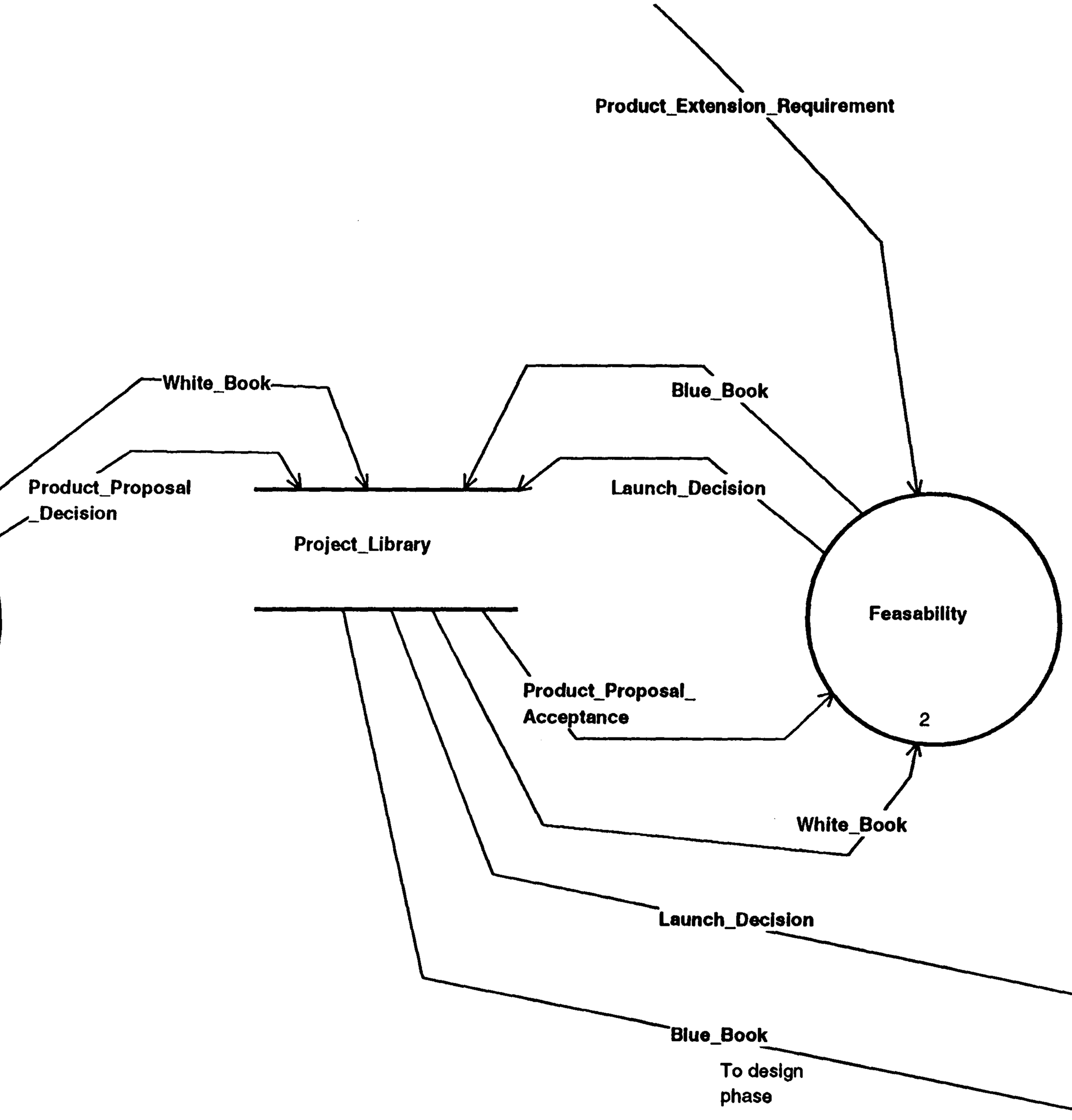
DFD 0 . . . . . Launch_Processes (5, 6, 7)
DFD 2 . . . . . Feasibility (5, 6, 7)
DFD 2.1 . . . . . Develop_Phase_Plan (3, 4, 5)
DFD 2.2 . . . . . Develop_Product_Proposal (3, 4)
DFD 2.3 . . . . . Develop_Project_Plan (3, 4)

Launch_Processes

Based on Templates
(Augmented from docs)

PHASES TWO & THREE

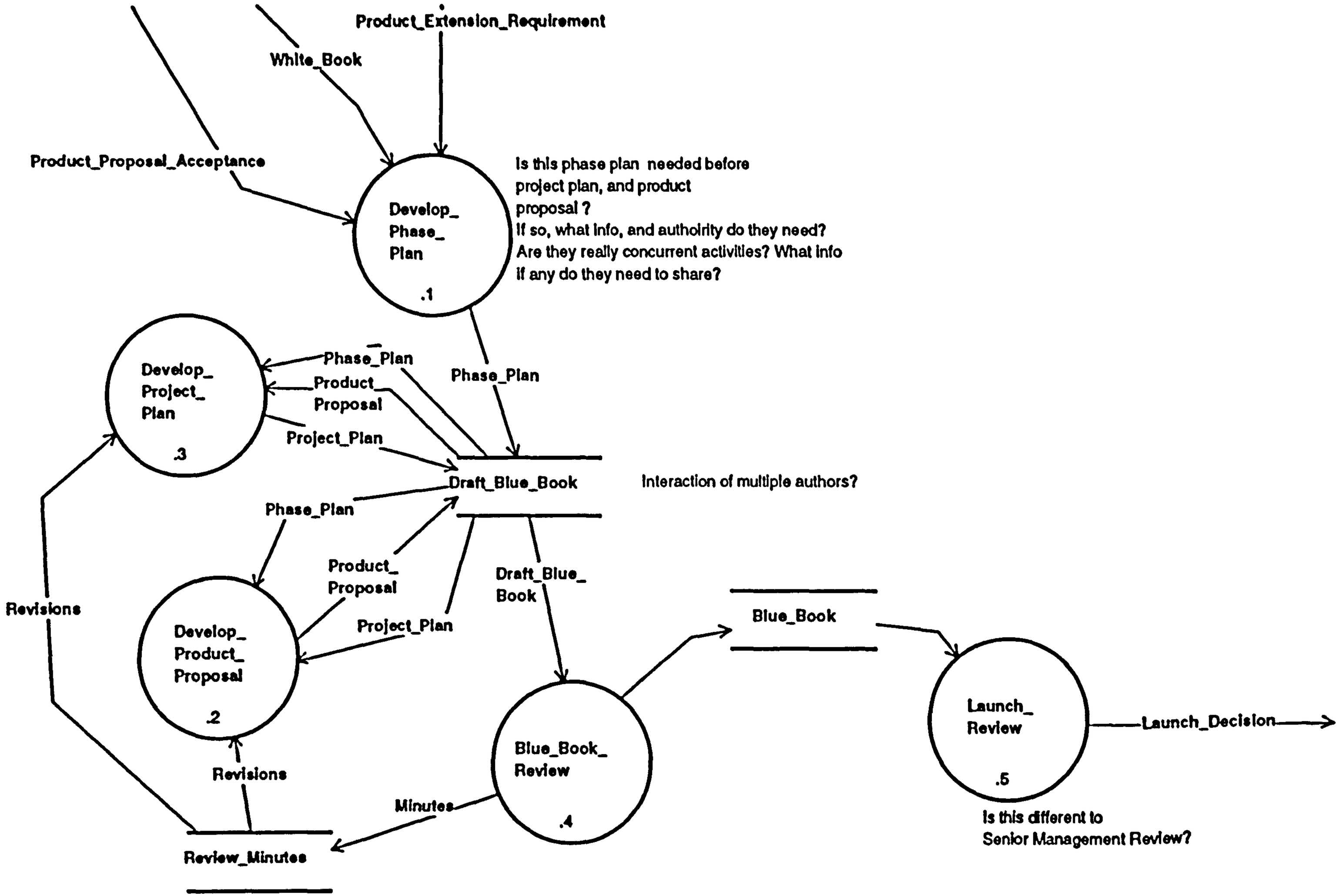




The repositry for docs was a guess at this stage.

:7
easibility

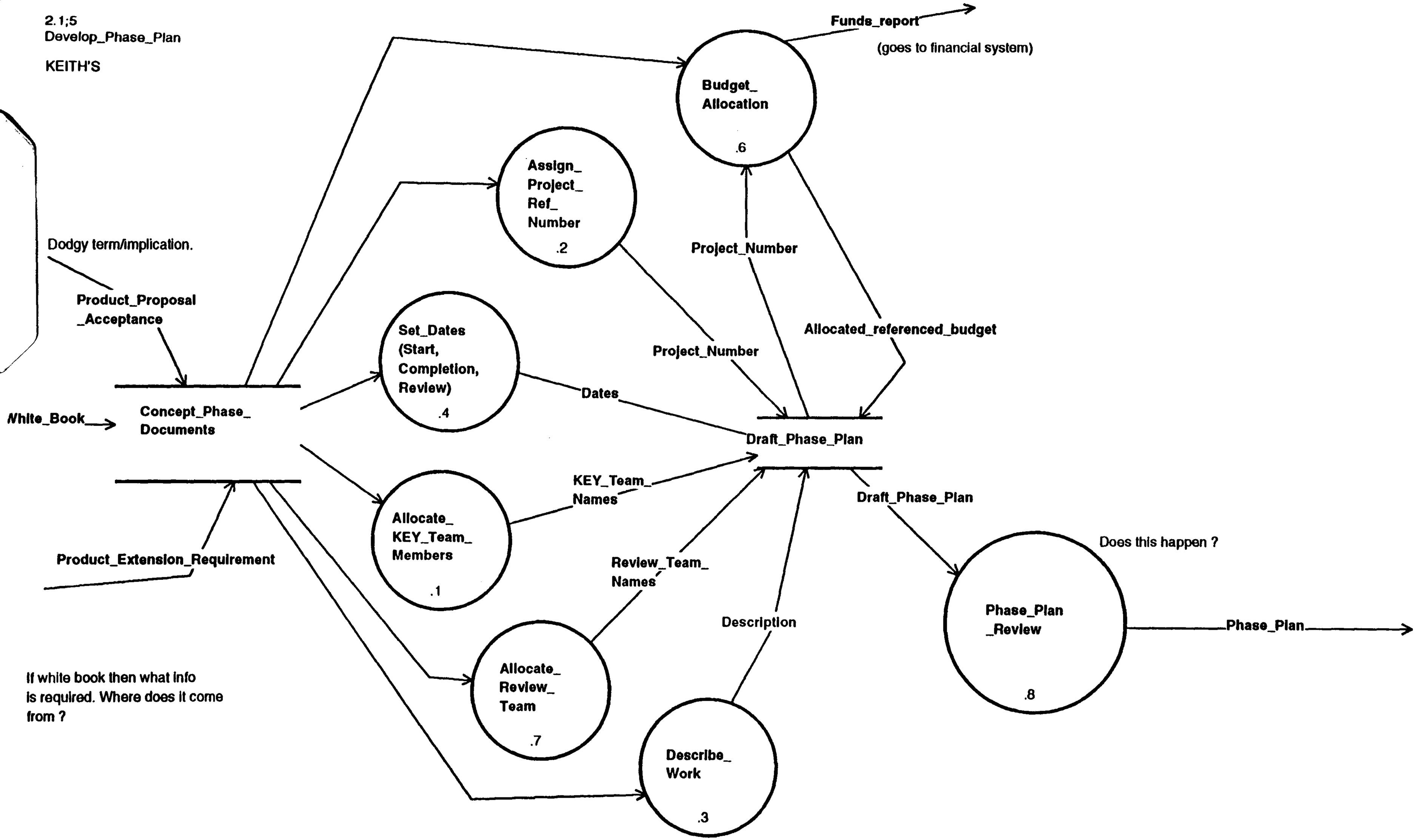
KEITH'S



Link to Project Control?

2.1;5
Develop_Phase_Plan

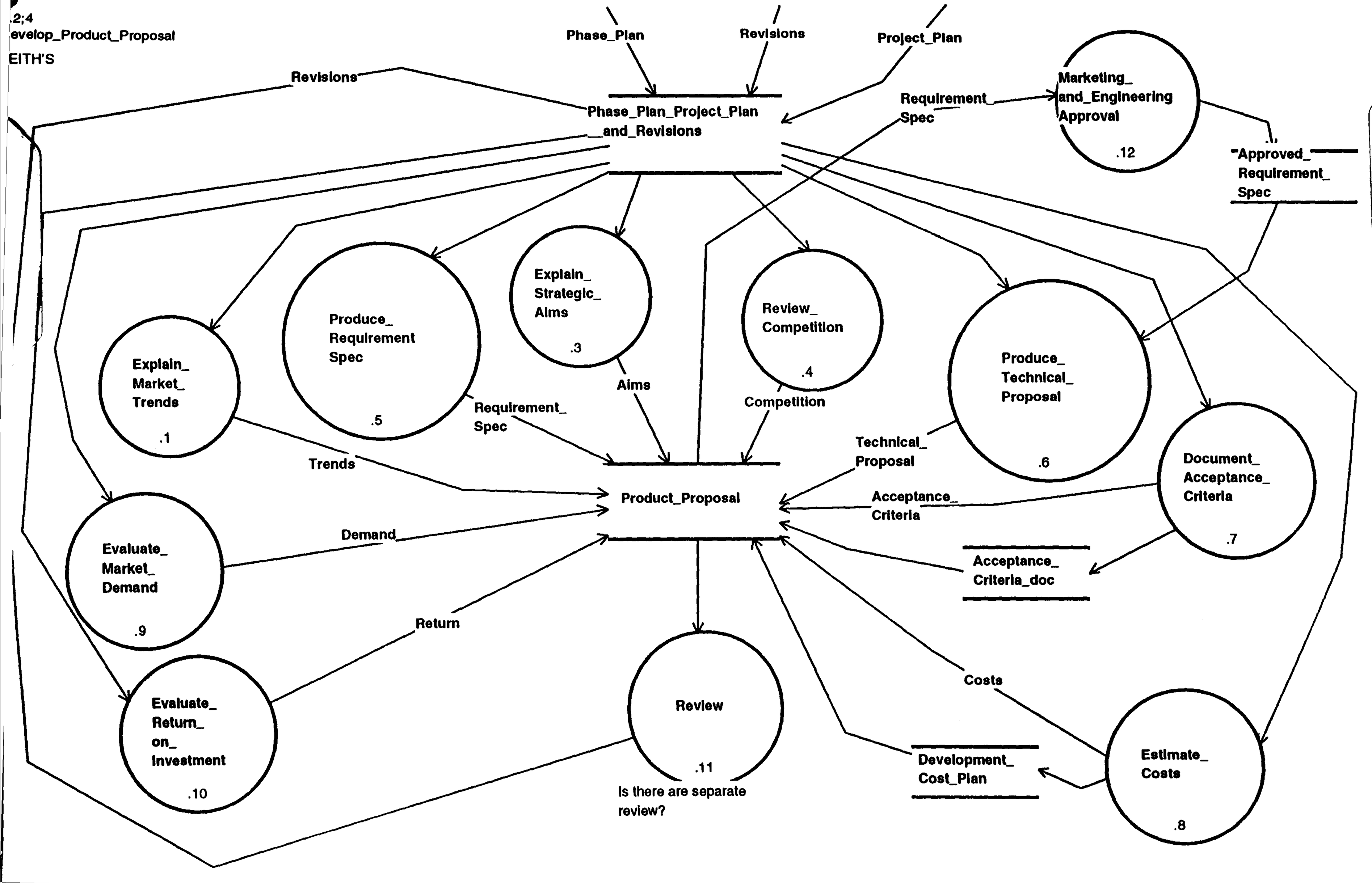
KEITH'S



If white book then what info is required. Where does it come from ?

TEXT CUT OFF IN ORIGINAL

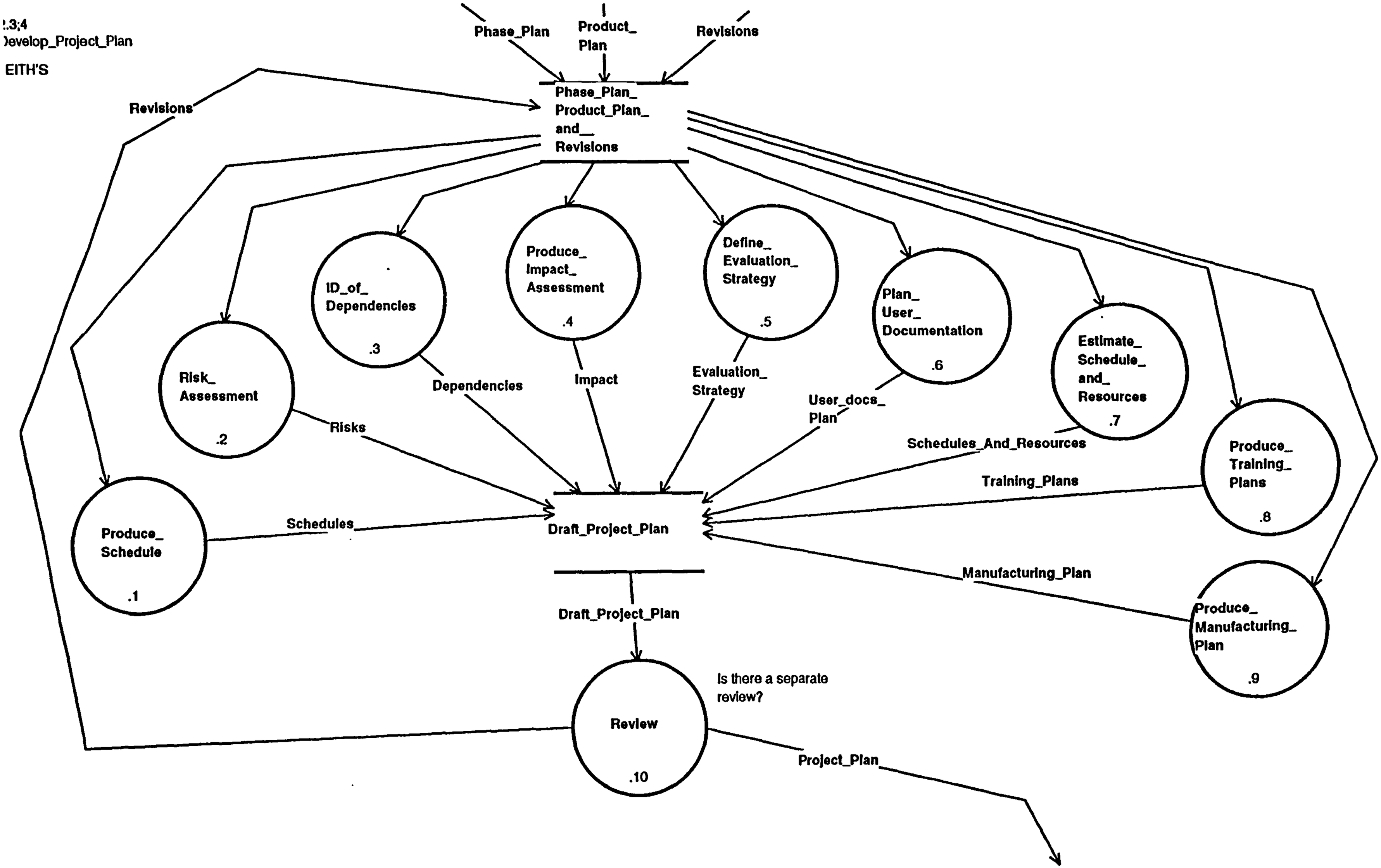
.2;4
evelop_Product_Proposal
EITH'S



Which of these (or parts of these) is needed?

!.3;4
)evelop_Project_Plan

EITH'S

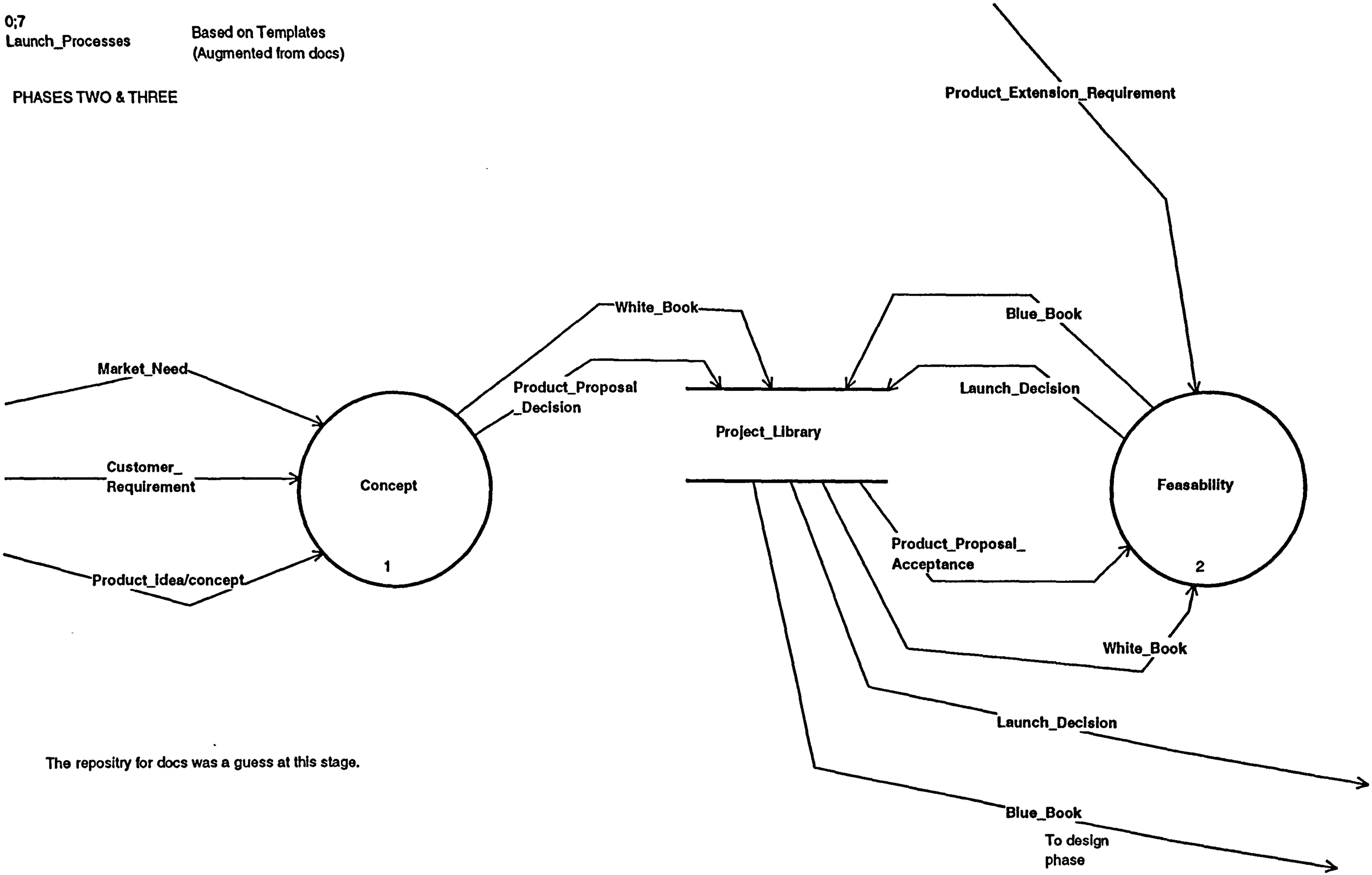


DFD 0 . . . . . Launch_Processes (5, 6, 7)
DFD 2 . . . . . Feasibility (5, 6, 7)
DFD 2.1 . . . . . Develop_Phase_Plan (3, 4, 5)
DFD 2.2 . . . . . Develop_Product_Proposal (3, 4)
DFD 2.3 . . . . . Develop_Project_Plan (3, 4)

0;7
Launch_Processes

Based on Templates
(Augmented from docs)

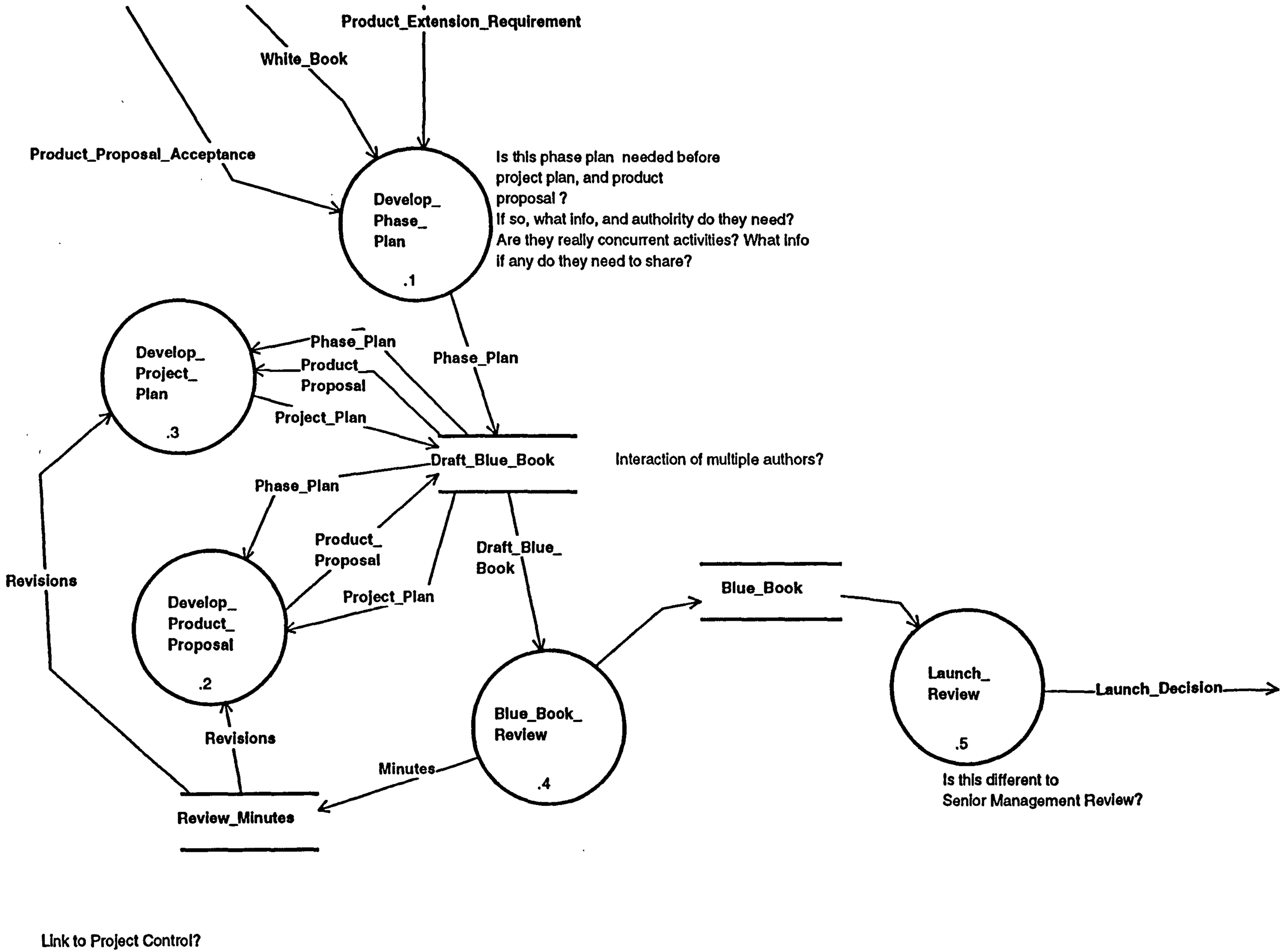
PHASES TWO & THREE



The repository for docs was a guess at this stage.

;7
easibility

<EITH'S



Link to Project Control?

2.1;5
Develop_Phase_Plan

KEITH'S

Budget_
Allocation
.6

Funds_report

(goes to financial system)

Assign_
Project_
Ref_
Number
.2

Dodgy term/implication.

Product_Proposal
_Acceptance

Project_Number

Set_Dates
(Start,
Completion,
Review)
.4

Project_Number

Allocated_referenced_budget

White_Book

Concept_Phase_
Documents

Dates

Draft_Phase_Plan

KEY_Team_
Names

Allocate_
KEY_Team_
Members
.1

Draft_Phase_Plan

Does this happen ?

Product_Extension_Requirement

Review_Team_
Names

Description

Phase_Plan
_Review
.8

Phase_Plan

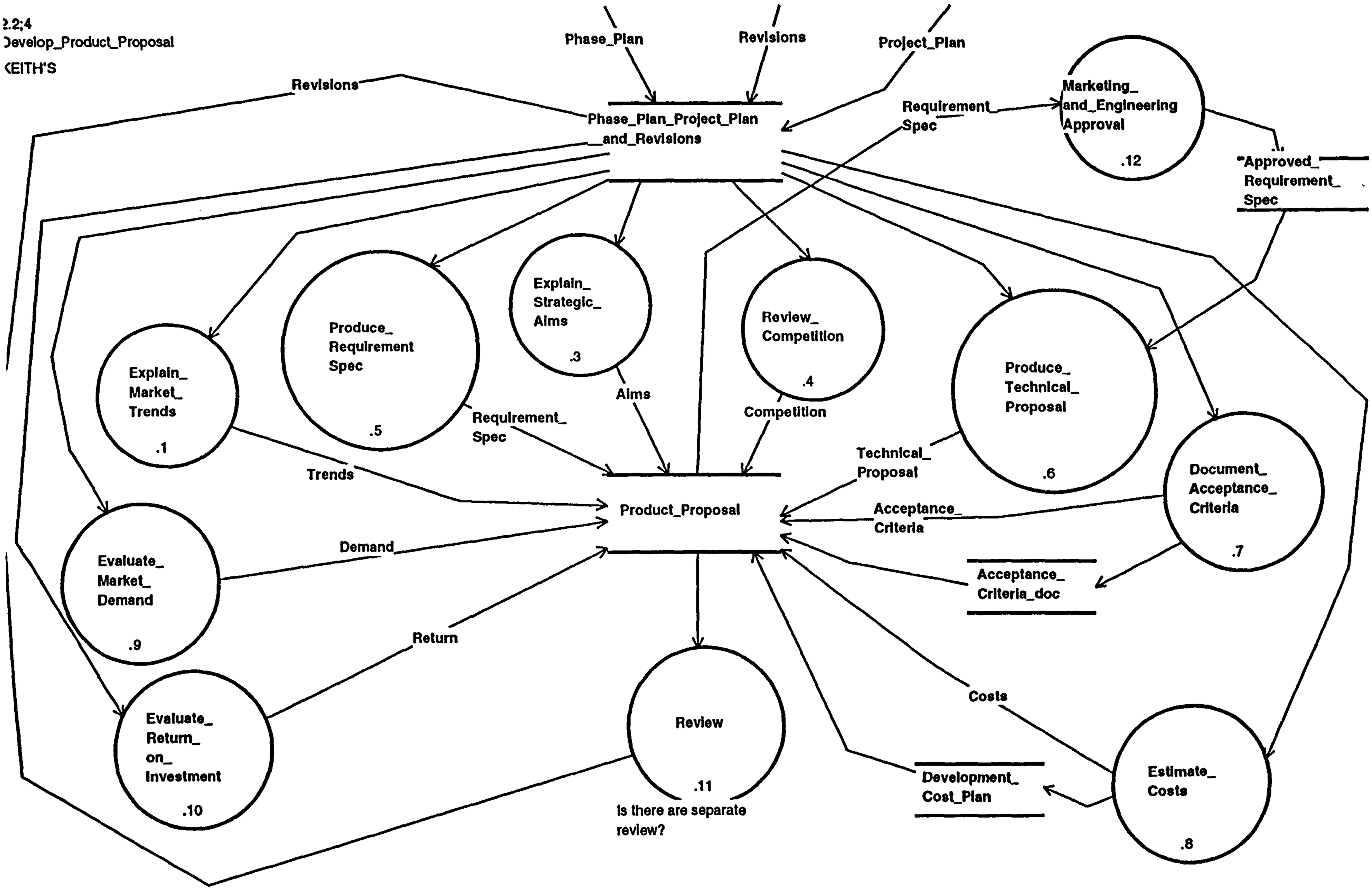
If white book then what info
is required. Where does it come
from ?

Allocate_
Review_
Team
.7

Describe_
Work
.3

Which of these (or parts of these) is needed?

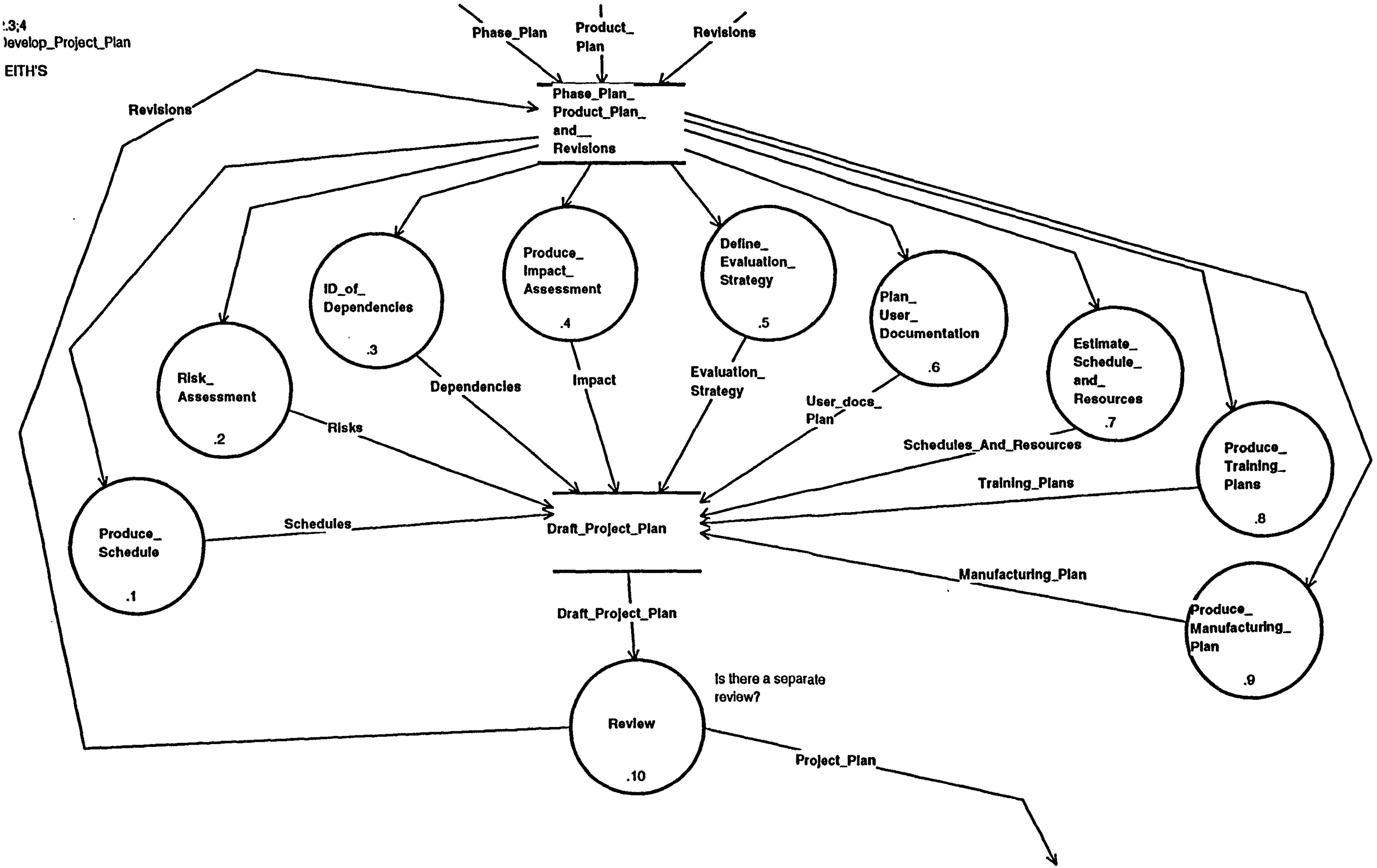
2.2;4
Develop_Product_Proposal
KEITH'S



Which of these (or parts of these) is needed?

:3;4
)evelop_Project_Plan

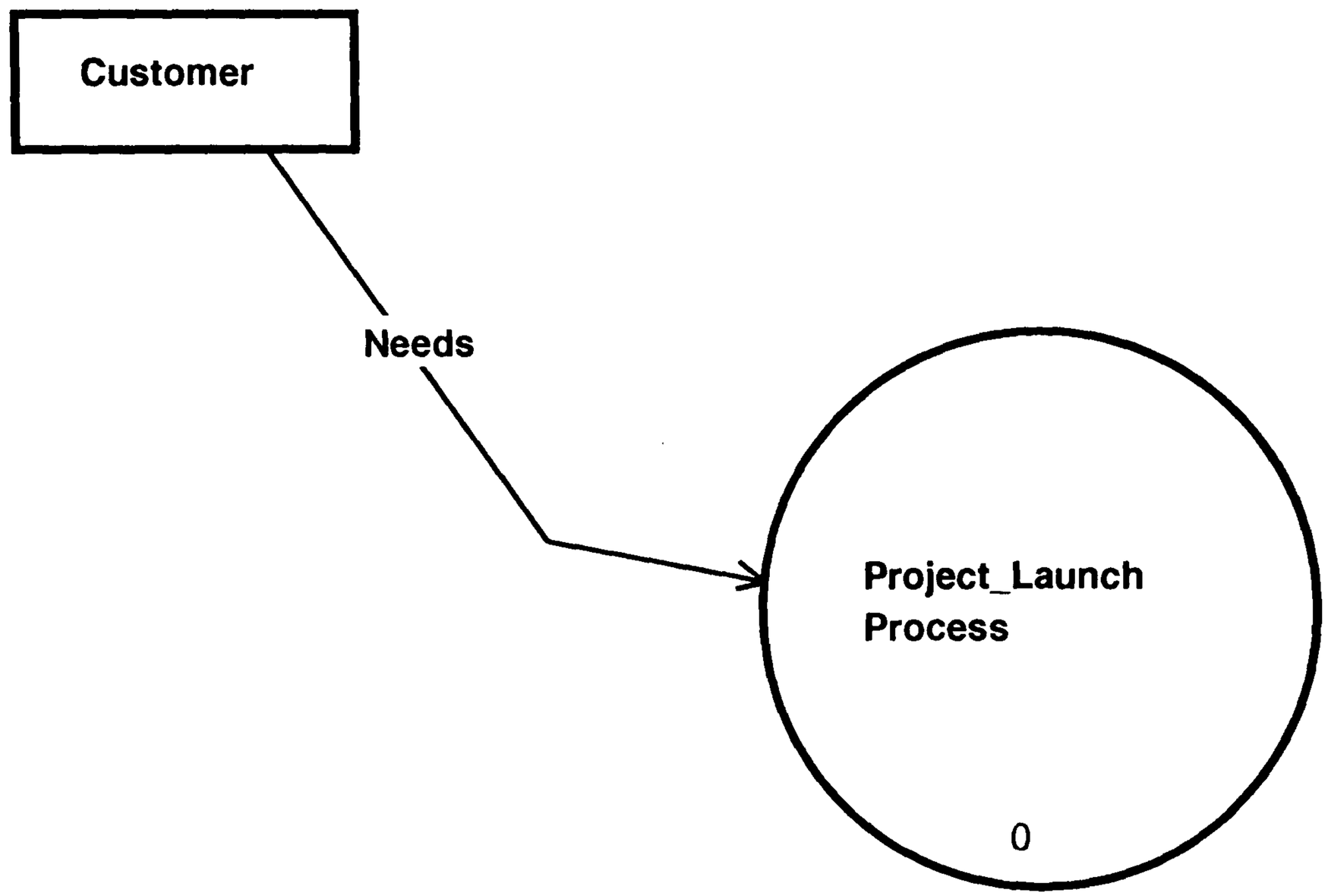
EITH'S



DFD	Context-Diagram	Actual_Launch_Process (1, 2)
DFD	0	Project_Launch Process (1, 2, 3, 4, 5)
DFD	1	Identification_of Product_Need (1)
DFD	2	Concept (1, 2, 3)
DFD	2.1	Develop_Phase_Plan (1)
PS	2.2	Develop_Project_Plan (1)
PS	2.3	Develop_Product_Proposal (1)
DFD	3	Feasibility (1, 2, 3)
DFD	3.1	Develop_Phase_Plan (1)
DFD	3.1.1	Produce_Phase_Plan(Feasibility_Start) (1)
DFD	3.1.2	Produce_Phase_Plan(Continuation) (1)
DFD	3.2	Develop_Project_Plan (1, 2, 3)
DFD	3.3	Develop_Product_Proposal (1)
PS	3.3.12	Marketing_and_Engineering Approval (1)

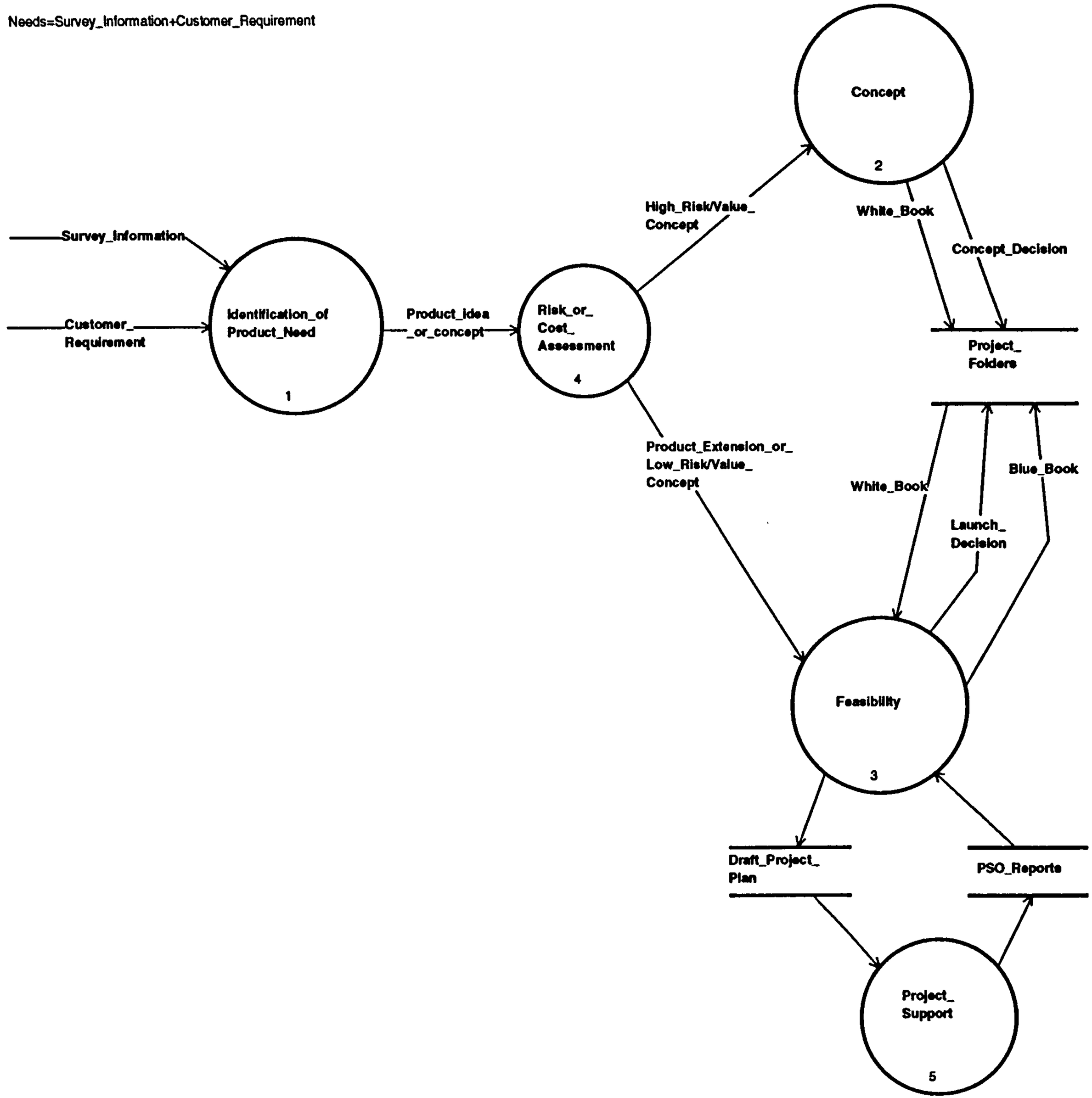
Context-Diagram;2
Actual_Launch_Process

PHASE FOUR



Project_Launch Process

Needs=Survey_Information+Customer_Requirement



:1
Identification_of Product_Need

Survey_Information

Identify_
Business_
Need

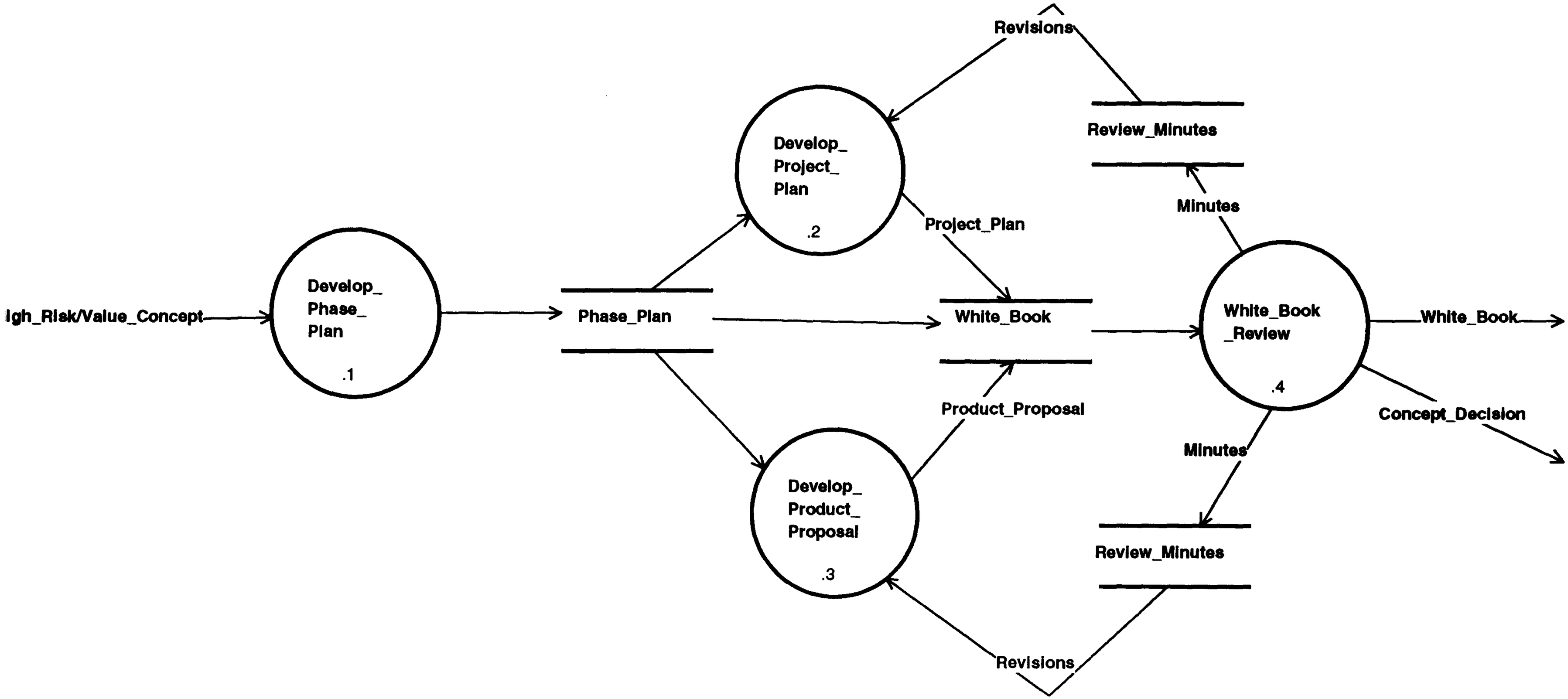
.1

Product_Idea_or_concept

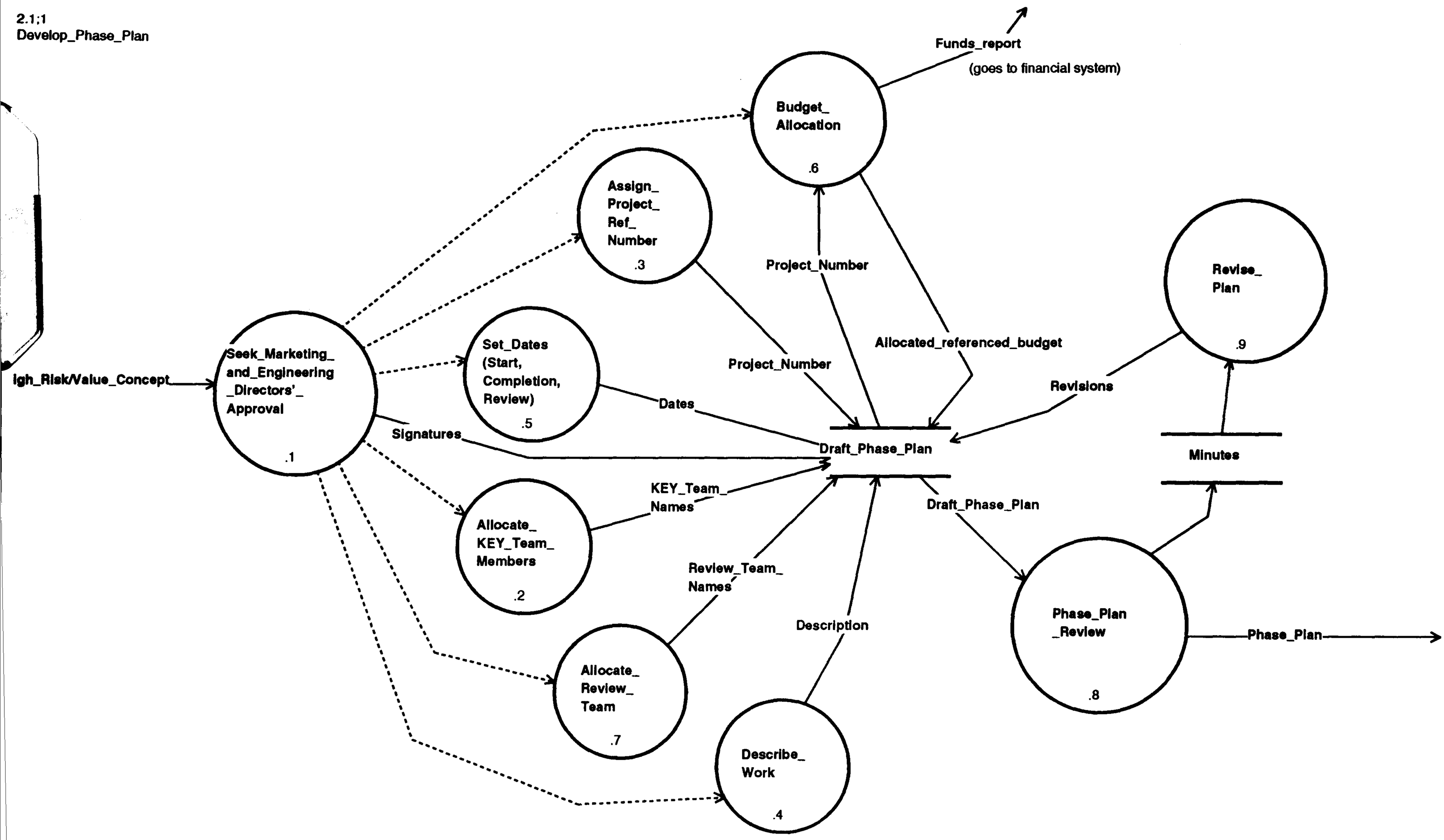
Customer_Requirement

Develop_
Product_
_Idea

.2


Revisions
Develop_
Project_
Plan
.2
Review_Minutes
Minutes
Project_Plan
Develop_
Phase_
Plan
.1
Phase_Plan
White_Book
White_Book
_Review
.4
White_Book
Concept_Decision
Product_Proposal
Minutes
Develop_
Product_
Proposal
.3
Review_Minutes
Revisions

2.1;1
Develop_Phase_Plan



NAME:
2.2;1

TITLE:
Develop_Project_Plan

INPUT/OUTPUT:
Revisions : data_in
Phase_Plan : data_in
Project_Plan : data_out

BODY:
This process is rarely carried out at the concept phase.
In theory it consists of the same activities as the Feasability Project Plan.
See Diagram for Feasability Project Plan for details.

NAME:
2.3;1

TITLE:
Develop_Product_Proposal

INPUT/OUTPUT:
Revisions : data_in
Phase_Plan : data_in
Product_Proposal : data_out

BODY:
This process is rarely carried out at the concept phase.
In theory it consists of the same activities as the Feasability Product Proposal.
See Diagram for Feasability Product Proposal for details.

3;3
Feasibility

White_Book

PSO_Reports

Draft_Project_Plan

Revisions_From_
Blue_Book_Review

Review_Minutes

Minutes

Product_Extension_or_
Low_Risk/Value_Concept

Develop_
Project_
Plan
.2

Project_Plan

Develop_
Phase_
Plan
.1

Phase_Plan

Draft_Blue_
Book

Blue__Book
_Review
.4

Blue_Book

Launch_Decision

Product_Proposal

Minutes

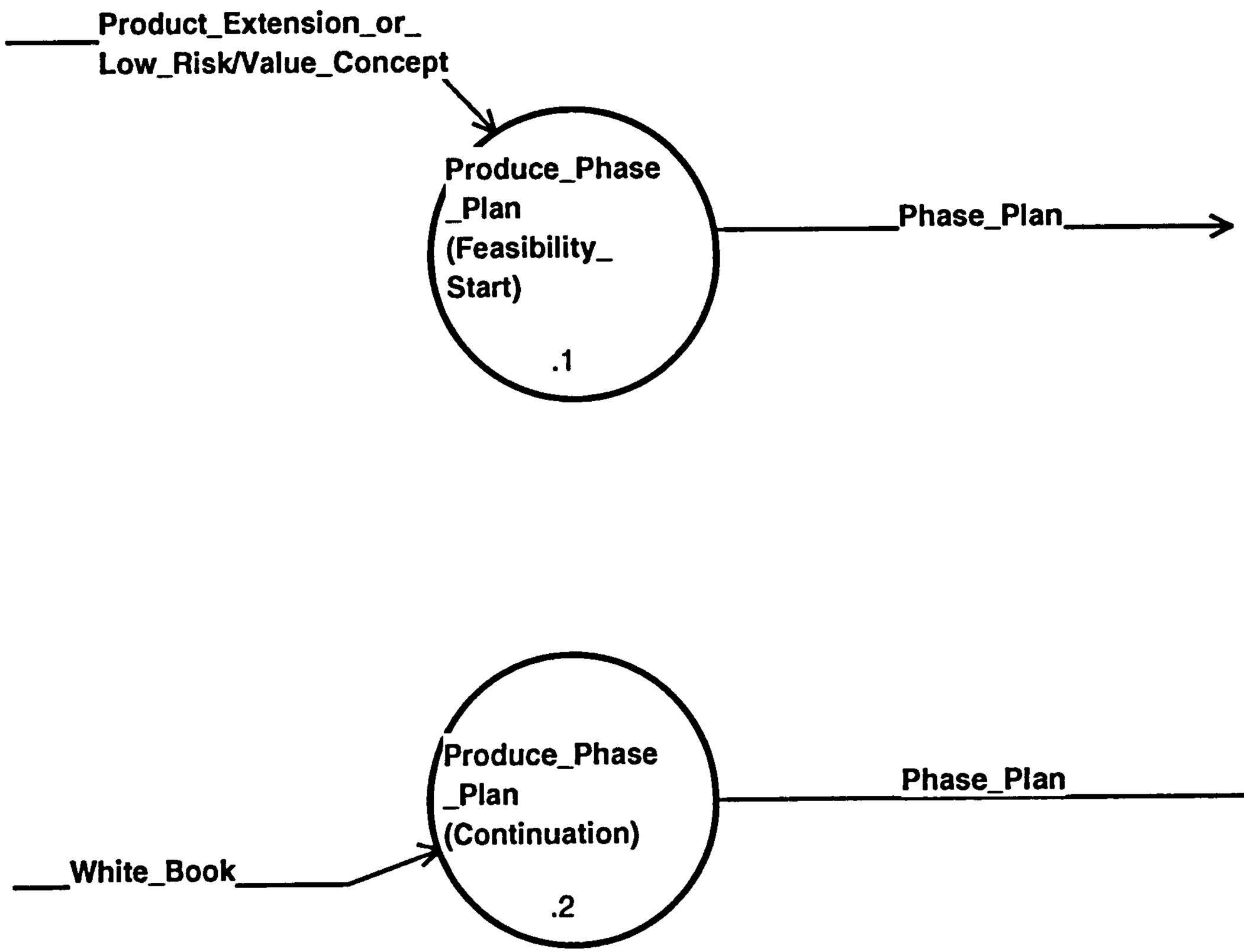
Develop_
Product_
Proposal
.3

White_Book

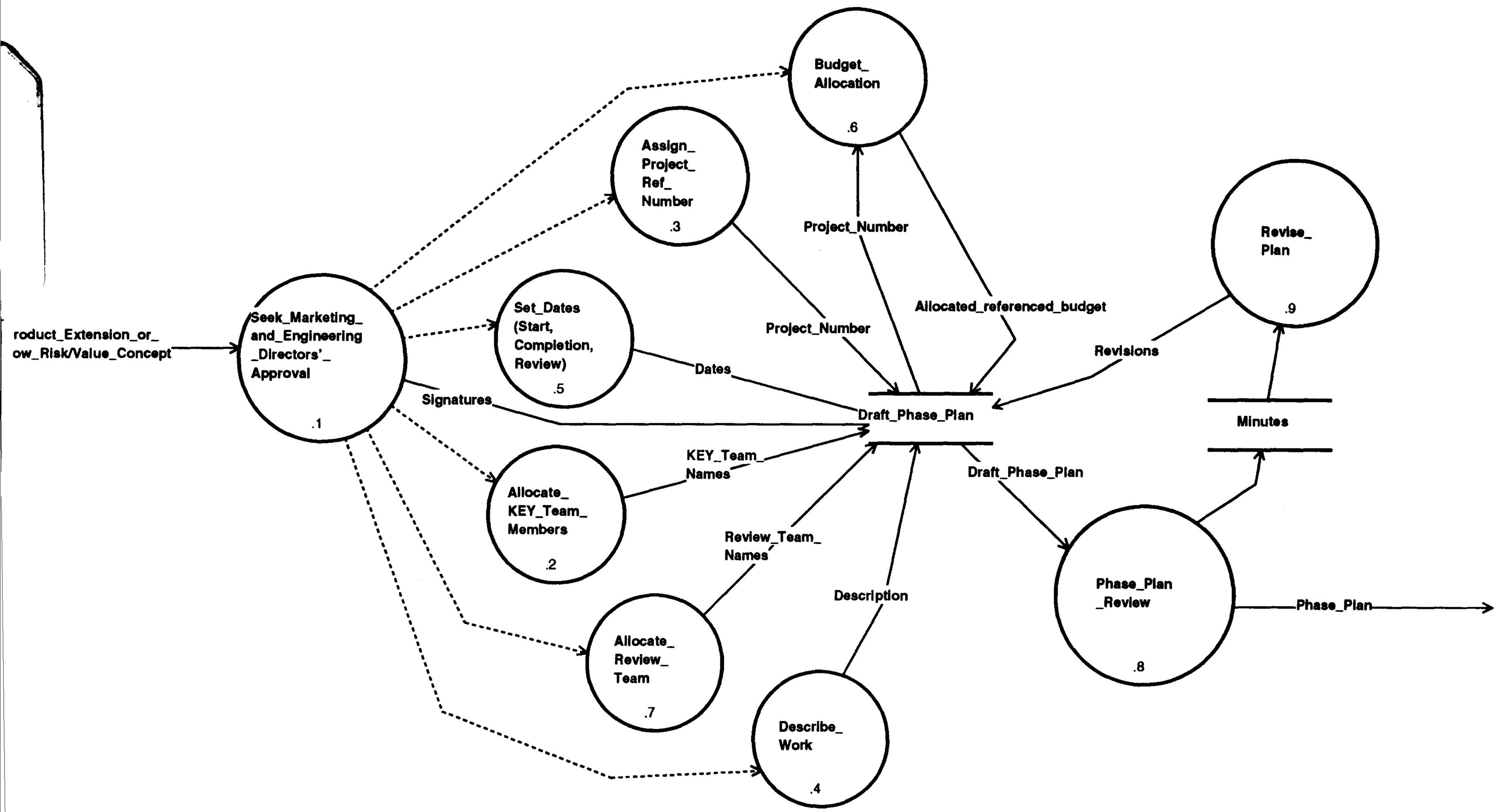
Review_Minutes

Revisions_From_
Blue_Book_Review

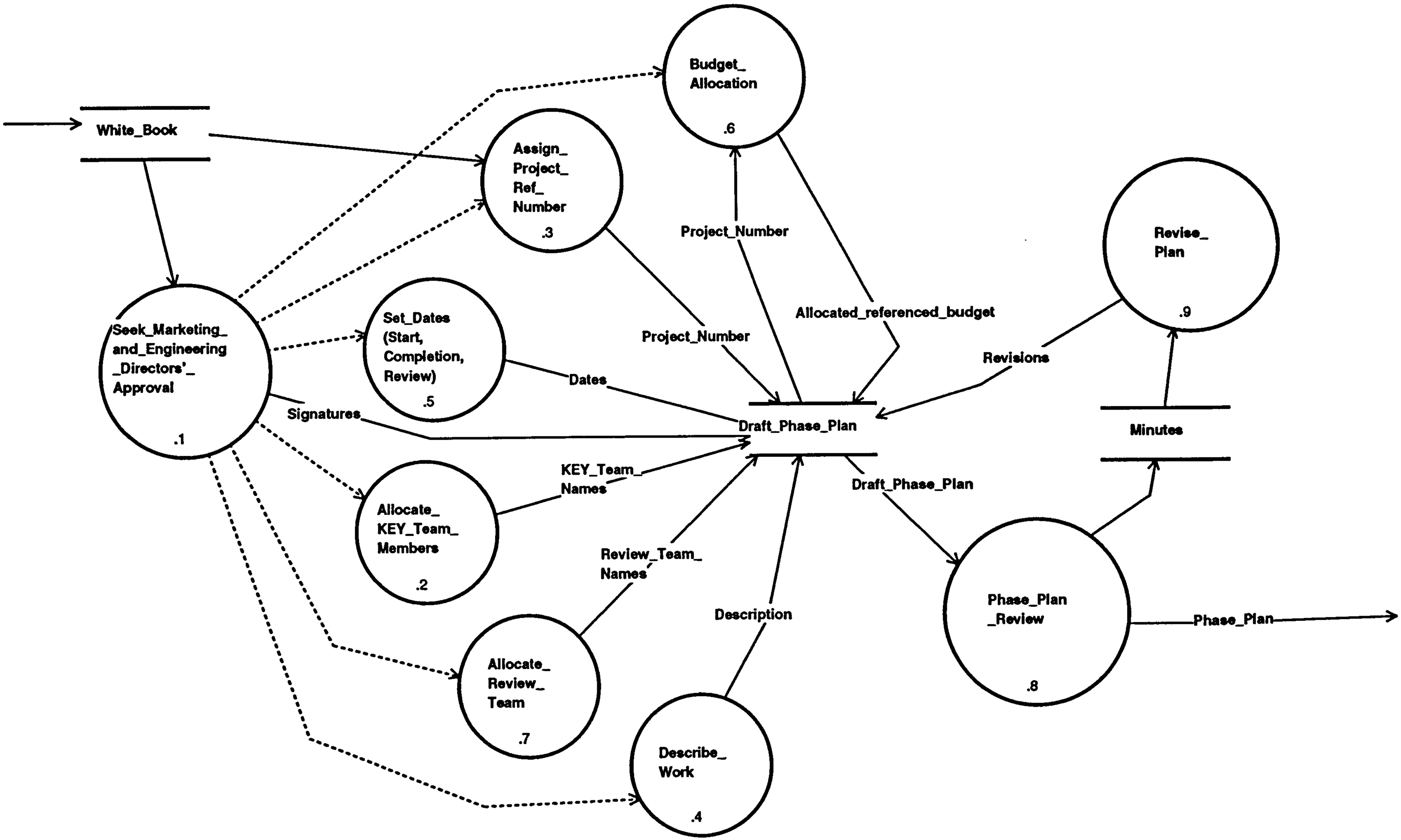
3.1;1
Develop_Phase_Plan



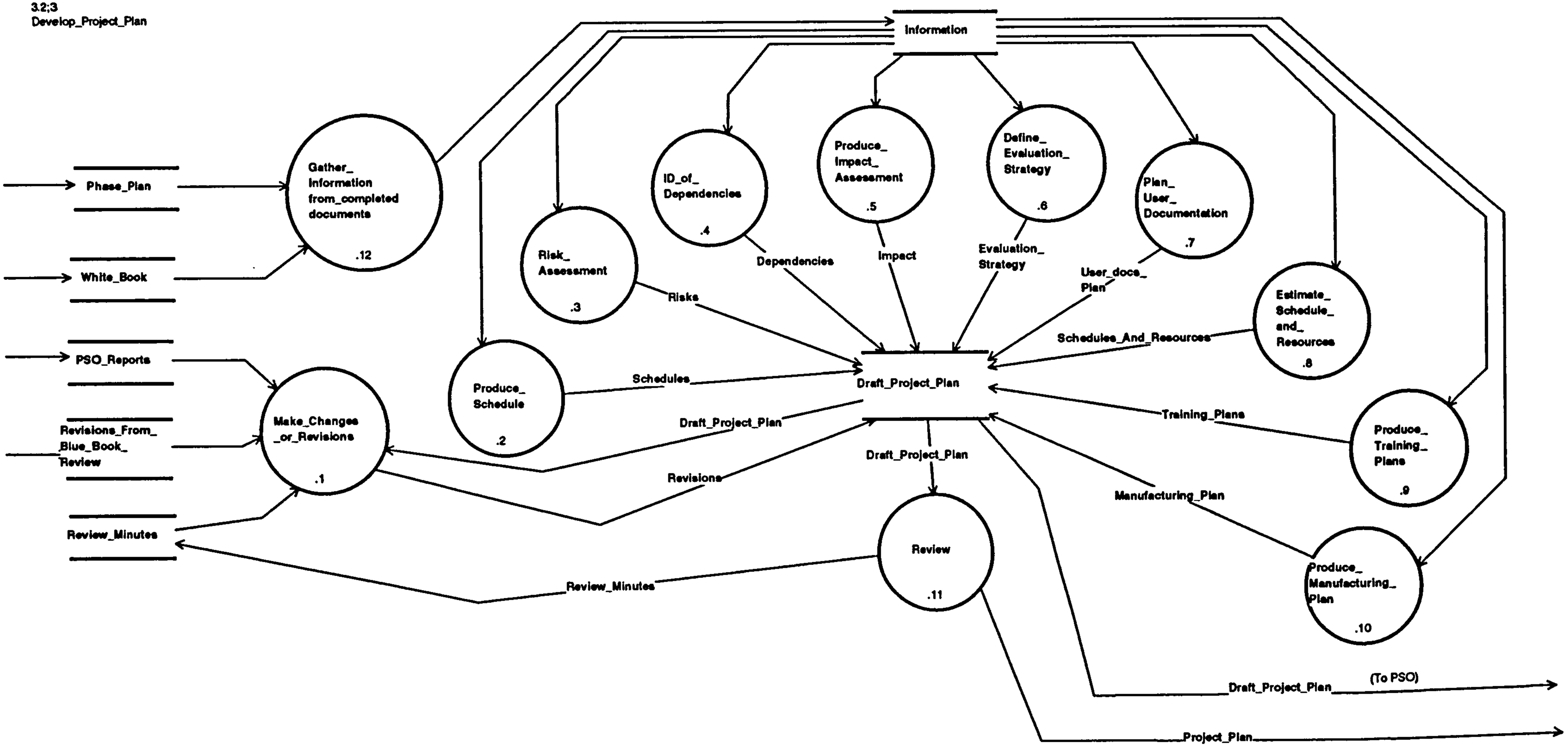
1.1;1
roduce_Phase_Plan(Feasibility_Start)



3.1.2;1
Produce_Phase_Plan(Continuation)



3.2;3
Develop_Project_Plan



;1
velop_Product_Proposal

n

Phase_Plan

White_Book

Gather_ Information from_completed documents .14

Information

Project_Plan

Requirement_ Spec

Marketing_ and_Engineering Approval .12

"Approved_ Requirement_ Spec

Explain_ Market_ Trends .1

Produce_ Requirement Spec .5

Explain_ Strategic_ Aims .3

Review_ Competition .4

Produce_ Technical_ Proposal .6

Document_ Acceptance_ Criteria .7

Aims

Competition

Requirement_ Spec

Technical_ Proposal

Trends

Product_Proposal

Acceptance_ Criteria

Revisions_From_ Blue_Book_Review

Evaluate_ Market_ Demand .9

Demand

Acceptance_ Criteria_doc

Return

Make_Changes _or_Revisions .13

Evaluate_ Return_ on_ Investment .10

Review .11

Costs

Revisions

Developement_ Cost_Plan

Estimate_ Costs .8

Review_Minutes

Product_Proposal

NAME:
3.3.12;1

TITLE:
Marketing_and_Engineering Approval

INPUT/OUTPUT:
Requirement_Spec : data_in
Approved_Requirement_Spec : data_out

BODY:
The template hidden text suggests that this is a definite sequence of events which must take place.

In reality this is not always the case. Sometimes the technical reply will be commenced before the agreement of the spec.

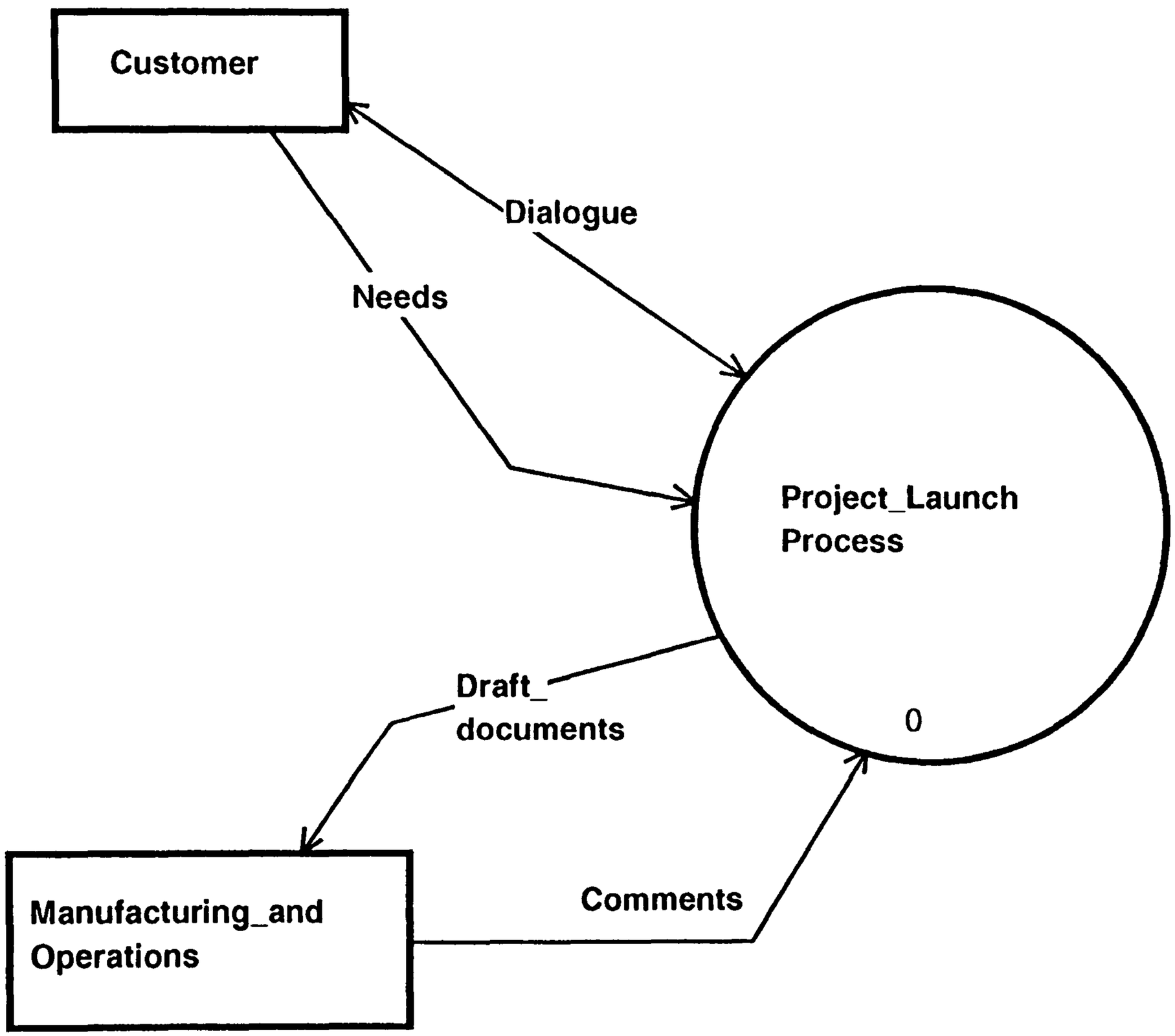
It is usual for the marketing rep to produce the requirements, which is then agreed before the engineering response. However, on other occasions these sections will be produced much more cooperatively.

The model hence really depicts an ideal, which may nevertheless still take place.

DFD	Context-Diagram	Actual_Launch_Process (1, 2, 3, 4)
DFD	0	Project_Launch Process (1, 2, 3, 4)
DFD	1	Identification_of Product_Need (1, 2)
DFD	2	Concept (1, 2, 3)
DFD	2.1	Develop_Phase_Plan (1, 2, 3, 4)
PS	2.2	Develop_Project_Plan (1, 2)
PS	2.3	Develop_Product_Proposal (1, 2)
DFD	3	Feasibility (1, 2, 3, 4, 5, 6)
DFD	3.1	Develop_Phase_Plan (1)
DFD	3.1.1	Produce_Phase_Plan(Feasibility_Start) (1, 2)
DFD	3.1.2	Produce_Phase_Plan(Continuation) (1, 2)
DFD	3.2	Develop_Project_Plan (1, 2, 3, 4, 5)
DFD	3.3	Develop_Product_Proposal (1, 2)
PS	3.3.12	Marketing_and_Engineering Approval (1, 2)
DFD	10	meta_model (1, 2)

Context-Diagram;4
Actual_Launch_Process

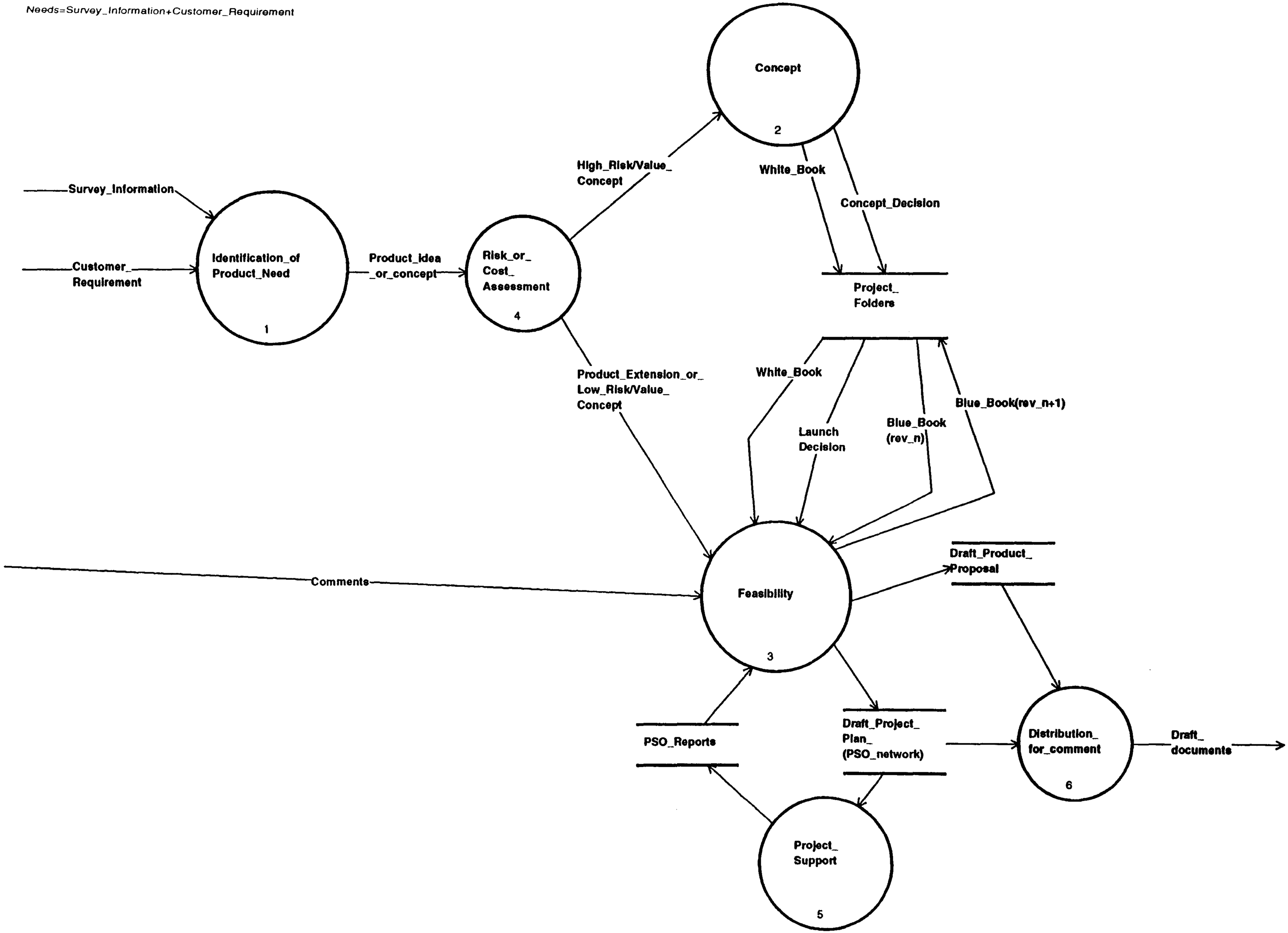
PHASE FIVE



Assumes that Engineering and Marketing are inside this system boundary

TEXT BOUND INTO THE SPINE

Needs=Survey_Information+Customer_Requirement



TITLE:
DFD Concepts

BODY:
PROCESS CONCEPT

Concept is an example of a process.

It has input High Risk / Value Concept and outputs of White Book and Concept Decision.

It can be seen in more detail (blown up) as diagram DFD 2 Concept.

Note that this again has the same inputs and outputs. This is the same process (Concept) seen in more detail. This ability to have a hierarchy of multiple levels of detail is one of the main advantages of the DFD approach.

The matching of the same inputs and outputs at the two levels is important. This means that the diagram is correctly 'balanced'.

STORES AND FLOWS

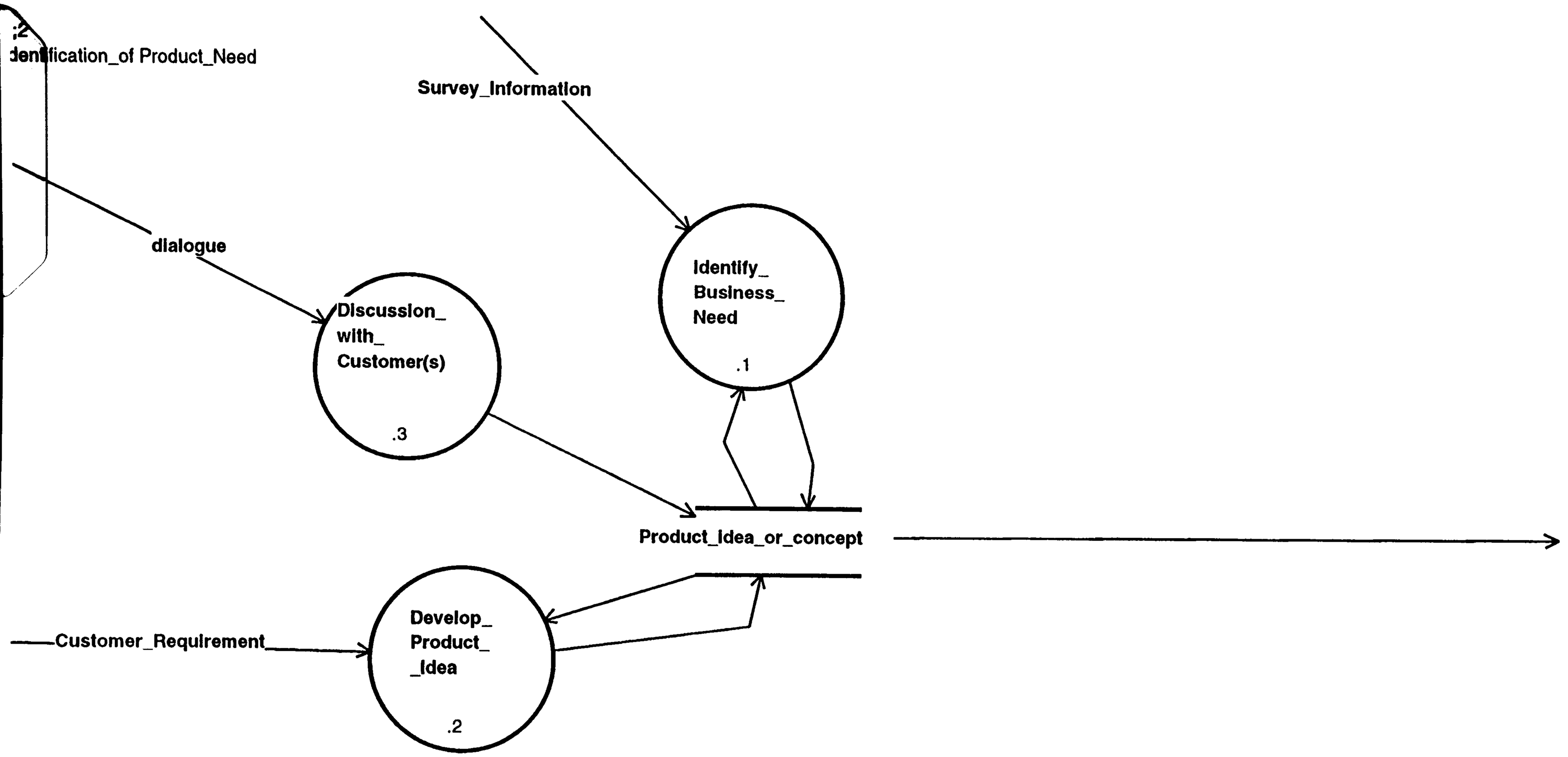
Stores include PSO reports, Project Folders etc. Stores hold information, they do not have to imply the phsyical storage medium.

Data flows (arrows) show the flow of information between processes (activities) and stores.

DATA COLLECTION

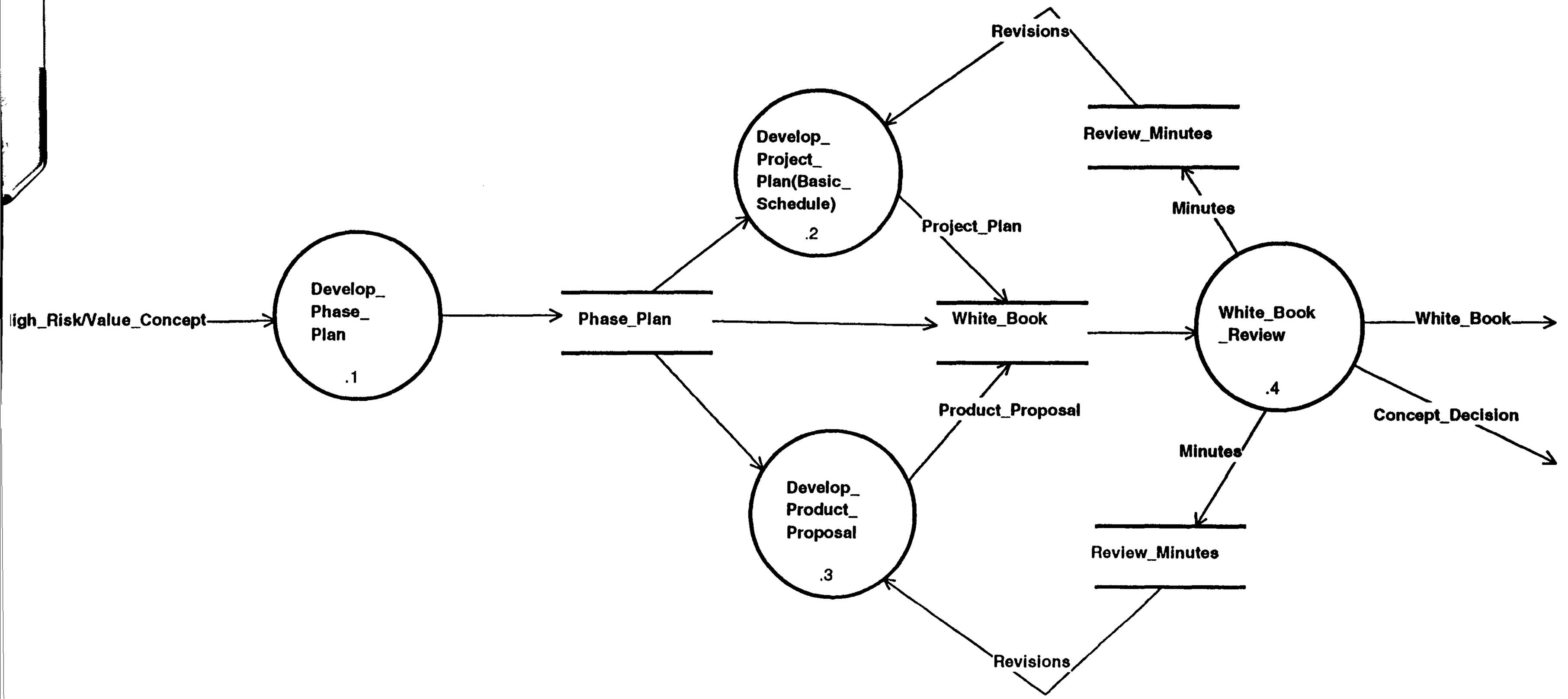
We can annotate the model with data collection points. For example we might collect the time at each iteration of the blue book. Our metric might be Time(Final_Revision(Signed off) - First Revision).

We may simply note the number of iterations, the duration of iterations etc.


dentification_of Product_Need
Survey_Information
dialogue
Discussion_
with_
Customer(s)
.3
Identify_
Business_
Need
.1
Product_Idea_or_concept
Customer_Requirement
Develop_
Product_
_Idea
.2

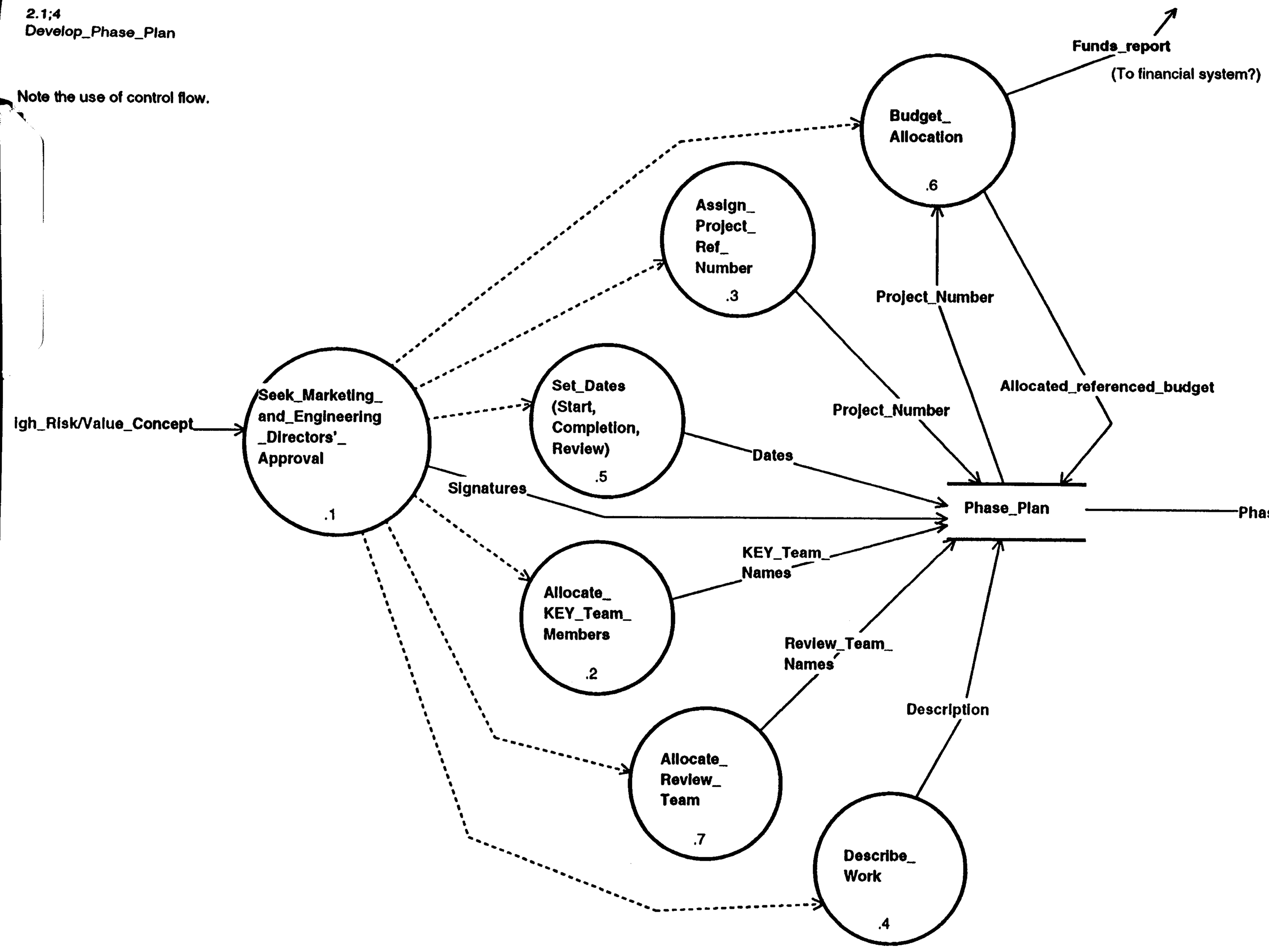
TEXT CUT OFF IN ORIGINAL

;3
ancept
ote that this is simply an expansion of the process
oncept from the level 0 DFD.



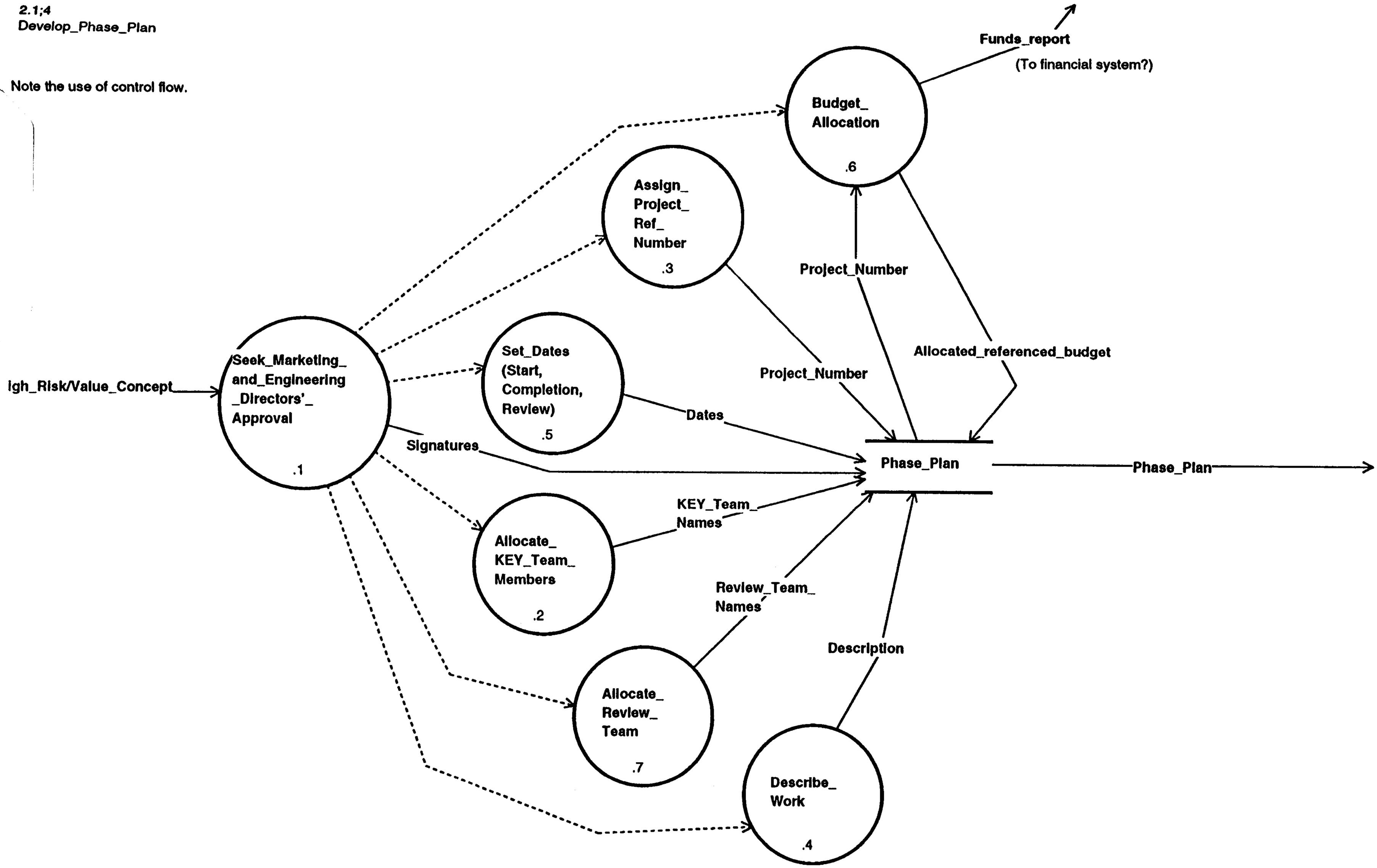
2.1;4
Develop_Phase_Plan

Note the use of control flow.



2.1;4
Develop_Phase_Plan

Note the use of control flow.



TITLE:
Control Flow

BODY:
Control flow is a way of enhancing DFDs with control or sequencing information.
The idea was developed to address real time issues (SASD becoming RT/SASD).
There are a number of variations on the theme (families), for example Hatley and Pirbhai, Ward and Mellor. I have used a simple variation of the idea, more Ward & Mellor, in which the flows simply activate a process.

The dotted line or control flow simply contains binary (eg on/off) information.

The implication here is that director approval is needed in order to carry out the other activities.

Is this true ?

The example will hopefully serve to illustrate that such enhancements can be used in order to force us to consider such sequncing issues, and to explicitly say when activities are sequenced, and when they may be concurrent.

NAME:
2.2;2

TITLE:
Develop_Project_Plan

INPUT/OUTPUT:
Revisions : data_in
Phase_Plan : data_in
Project_Plan : data_out

BODY:
This process is rarely carried out at the concept phase.
In theory it consists of the same activities as the Feasability Project Plan.
See Diagram for Feasability Project Plan for details.

In practice all that is required is a basic schedule.

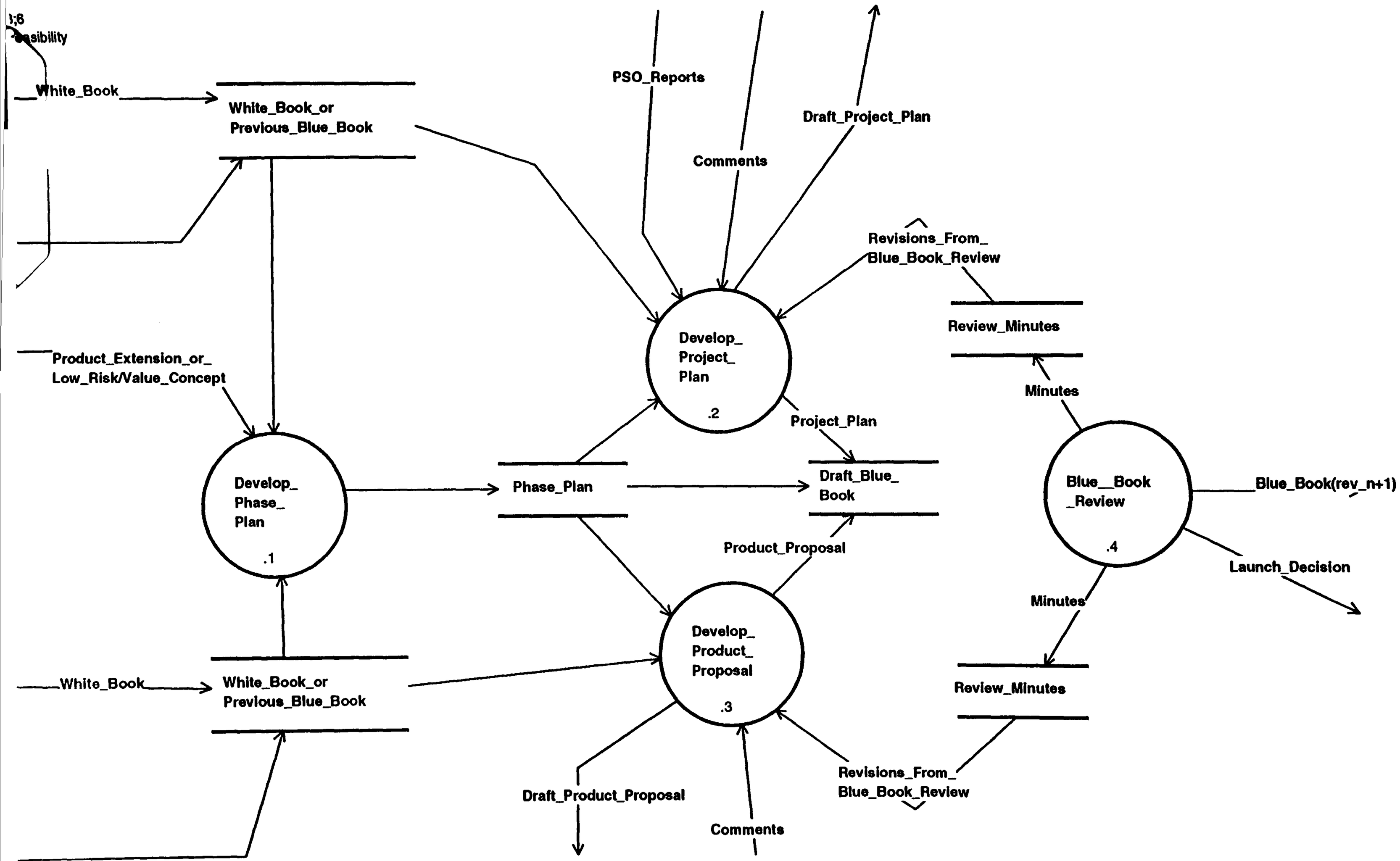
NAME:
2.3;2

TITLE:
Develop_Product_Proposal

INPUT/OUTPUT:
Revisions : data_in
Phase_Plan : data_in
Product_Proposal : data_out

BODY:
This process is rarely carried out at the concept phase.
In theory it consists of the same activities as the Feasability Product Proposal.
See Diagram for Feasability Product Proposal for details.

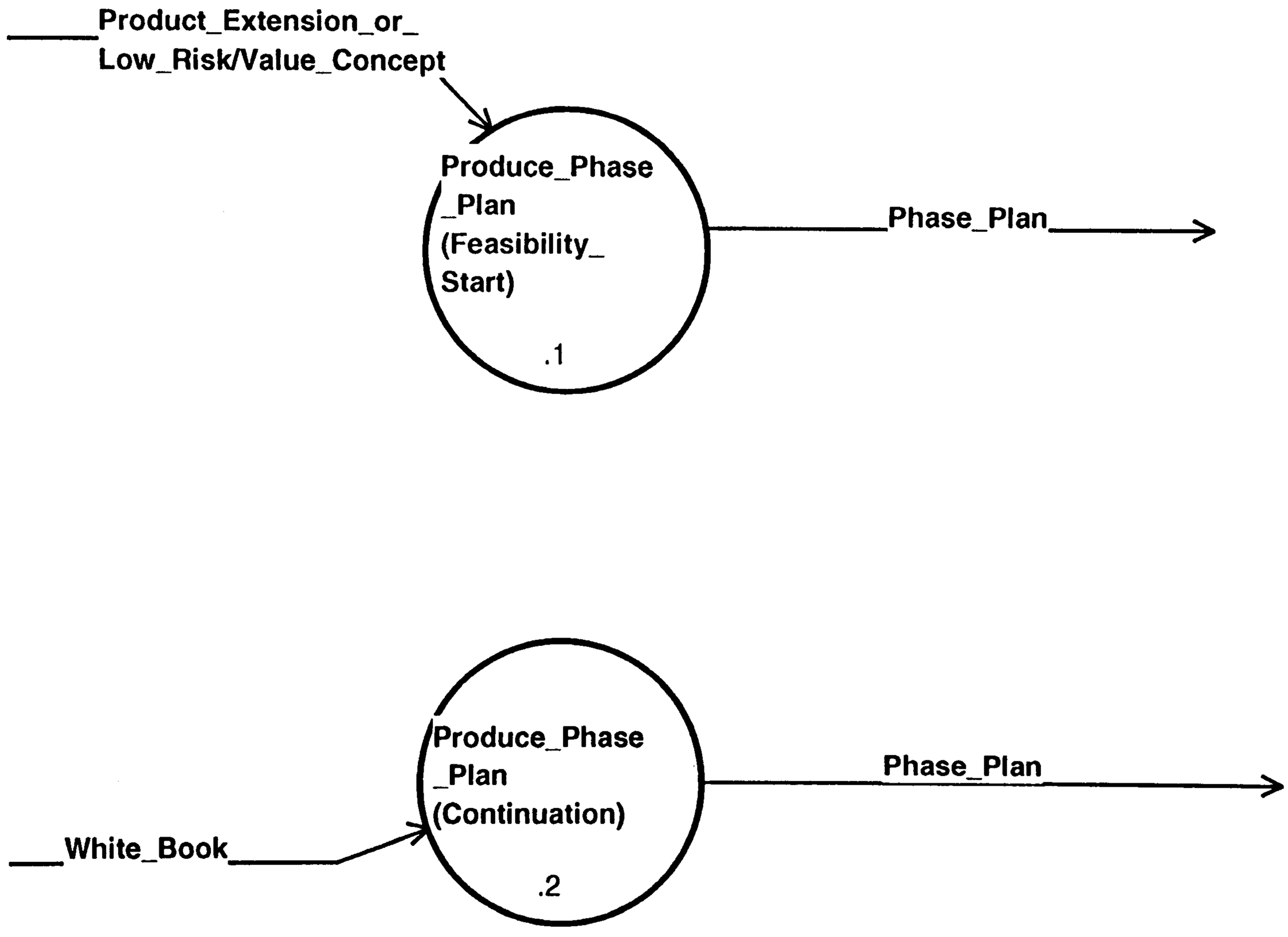
* PSpecs can be used (as above) to simply give textual guidance. However, they can be more formal,
for example using State Transition Diagrams, pseudo code, Structured English etc. *


1;6
asibility
White_Book
White_Book_or
Previous_Blue_Book
PSO_Reports
Comments
Draft_Project_Plan
Revisions_From_
Blue_Book_Review
Review_Minutes
Minutes
Develop_
Project_
Plan
.2
Project_Plan
Product_Extension_or_
Low_Risk/Value_Concept
Develop_
Phase_
Plan
.1
Phase_Plan
Draft_Blue_
Book
Blue__Book
_Review
.4
Blue_Book(rev_n+1)
Launch_Decision
Minutes
Product_Proposal
Develop_
Product_
Proposal
.3
White_Book
White_Book_or
Previous_Blue_Book
Review_Minutes
Draft_Product_Proposal
Comments
Revisions_From_
Blue_Book_Review

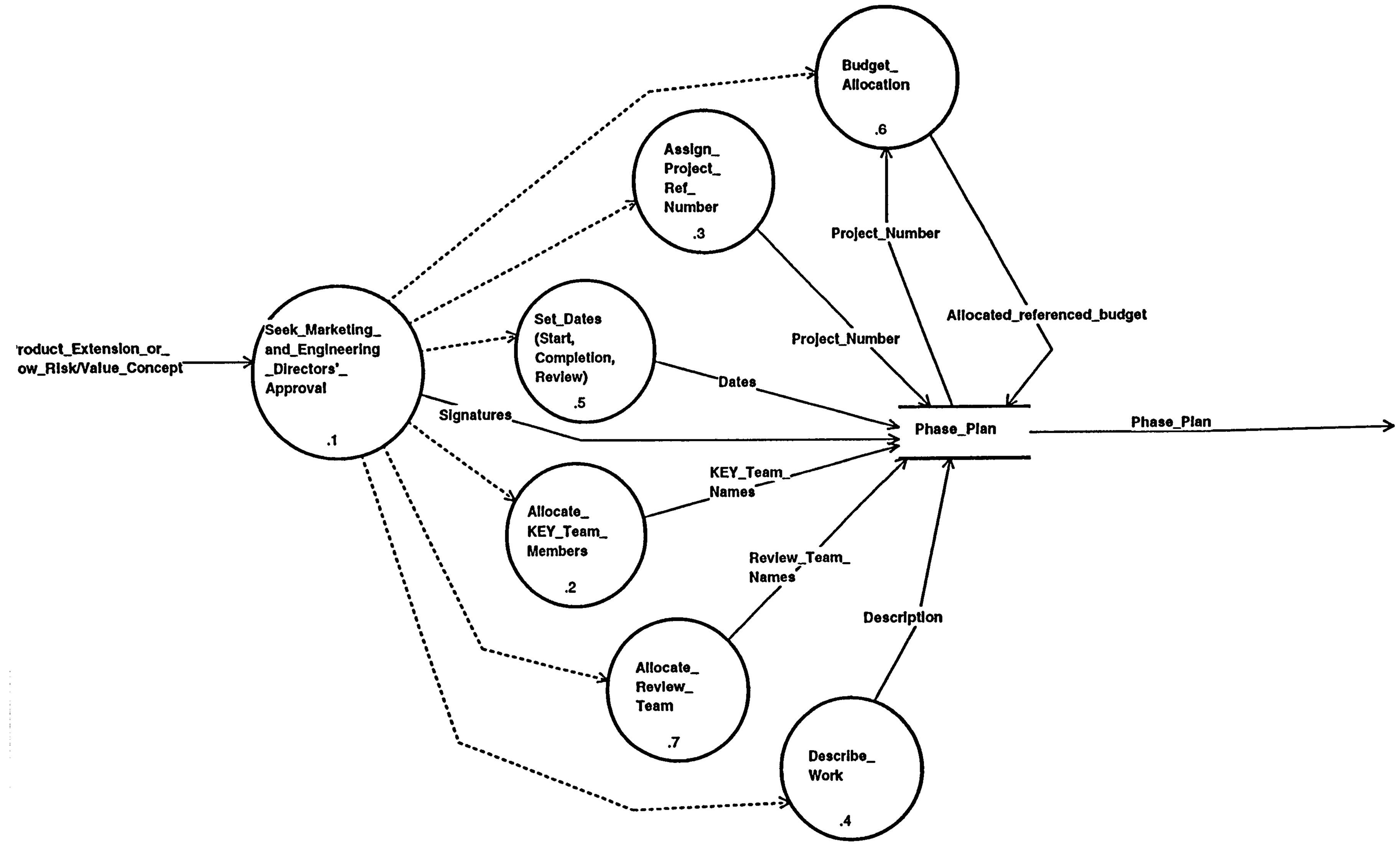
TITLE:
Feasibility Product Proposal

BODY:
This can be expanded to show different detail (number of processes), depending on the point in development, user etc. A number of alternative models could be produced in this way, by simply removing, processes from the complex version. (see DFD 3 and its expansion DFD 3.3)

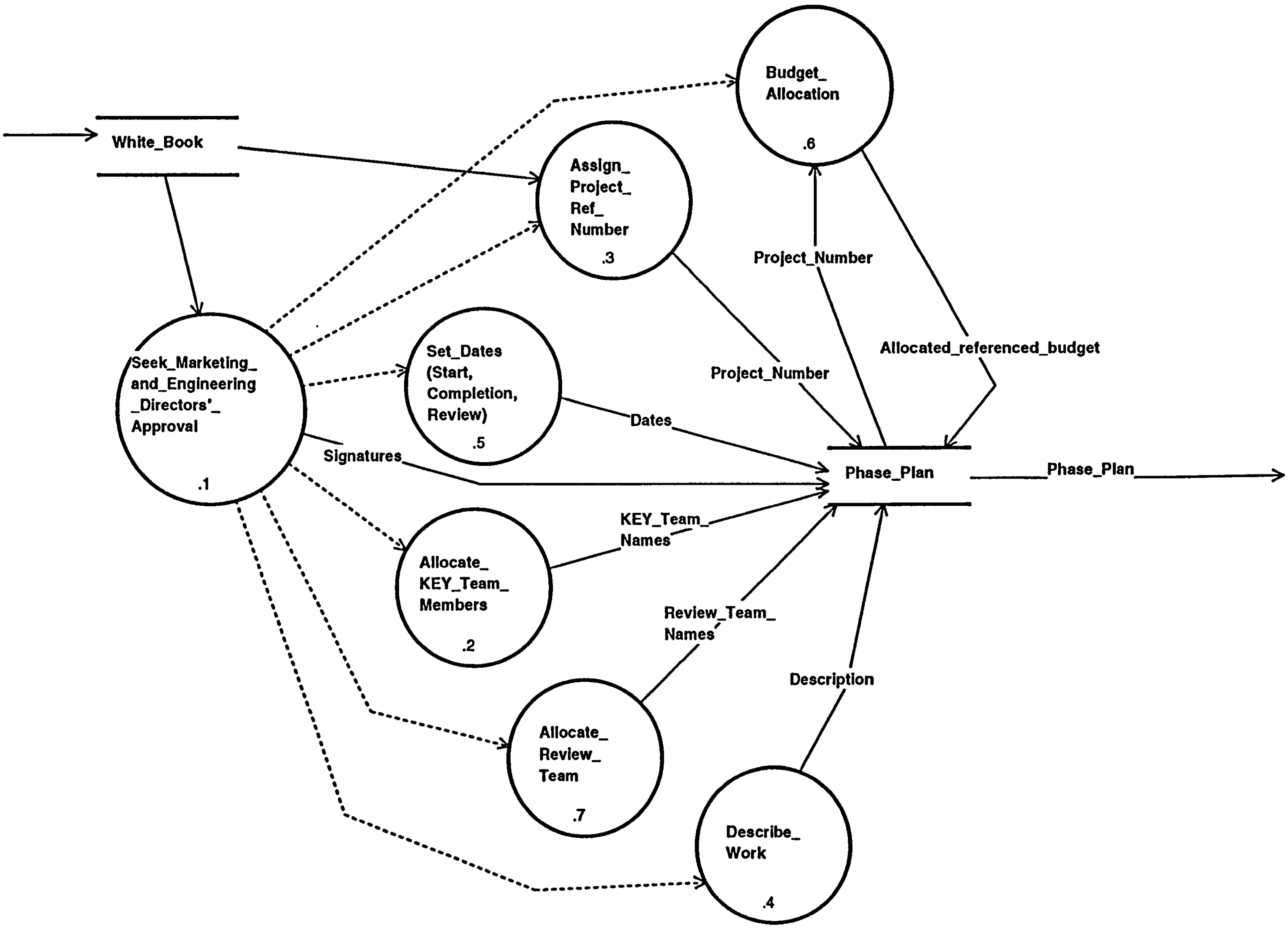
3.1;1
Develop_Phase_Plan



.1.1;2
roduce_Phase_Plan(Feasibility_Start)



3.1.2;2
Produce_Phase_Plan(Continuation)



2.5
Develop_Project_Plan

Phase_Plan

White_Book

(Unlikley before launch ?)

PSO_Reports

Revisions_From_ Blue_Book_ Review

Comments_from_ other_Business_ Functions

Review_Minutes

Gather_ Information from_completed documents .12

Make_Changes _or_Revisions .1

Risk_ Assessment .3

ID_of_ Dependencies .4

Produce_ Schedule .2

Produce_ Impact_ Assessment .5

Define_ Evaluation_ Strategy .6

Plan_ User_ Documentation .7

Estimate_ Schedule_ and_ Resources .8

Produce_ Training_ Plans .9

Produce_ Manufacturing_ Plan .10

Review .11

Create_ Work_ Packages .13

Information

Risks

Dependencies

Impact

Evaluation_ Strategy

User_docs_ Plan

Schedules_And_Resources

Schedules

Draft_Project_Plan

Training_Plans

Manufacturing_Plan

Work_Package

Project_Plan

(To PSO, Manufacturing, Operations, etc for comment)

Revisions

Review_Minutes

.3;2
levelop_Product_Proposal

Gather_ Information from_completed documents .14

Phase_Plan

White_Book

Revisions_From_ Blue_Book_Review

Comments_from_ other_Business_ Functions

Review_Minutes

Make_Changes _or_Revisions .13

Information

Project_Plan

Requirement_ Spec

Marketing_ and_Engineering Approval .12

"Approved_ Requirement_ Spec

Explain_ Market_ Trends .1

Produce_ Requirement Spec .5

Explain_ Strategic_ Aims .3

Review_ Competition .4

Produce_ Technical_ Proposal .6

Document_ Acceptance_ Criteria .7

Requirement_ Spec

Aims

Competition

Technical_ Proposal

Trends

Product_Proposal

Acceptance_ Criteria

Acceptance_ Criteria_doc

Evaluate_ Market_ Demand .9

Demand

Return

Evaluate_ Return_ on_ Investment .10

Review .11

Revisions

Costs

Developement_ Cost_Plan

Estimate_ Costs .8

Product_Proposal

Draft_Product_Proposal (for comment)

NAME:
3.3.12;3

TITLE:
Marketing_and_Engineering Approval

INPUT/OUTPUT:
Requirement_Spec : data_in
Approved_Requirement_Spec : data_out

BODY:
The template hidden text suggests that this is a definite sequence of events which must take place.

In reality this is not always the case. Sometimes the technical reply will be commenced before the agreement of the spec.

It is usual for the marketing rep to produce the requirements, which is then agreed before the engineering response. However, on other occasions these sections will be produced much more cooperatively.

The model hence really depicts an ideal, which may nevertheless still take place.

* The important point to note is that we can add appropriate guidance to the model where we wish. *

Comments_from_other_Business_Functions (store) =

This may be comments received from :

Customer (via Marketing)

Manufacturing

Operations etc.

This has been separated from PSO comment because it does not have a formal mechanism. The PSO mechanism is described by procedures, whereas the above are often informal. However, as with PSO, this may go through a number of iterations.

Also note that no explict connection to marketing is shown. This is because it is implict in the process, and internal to it.

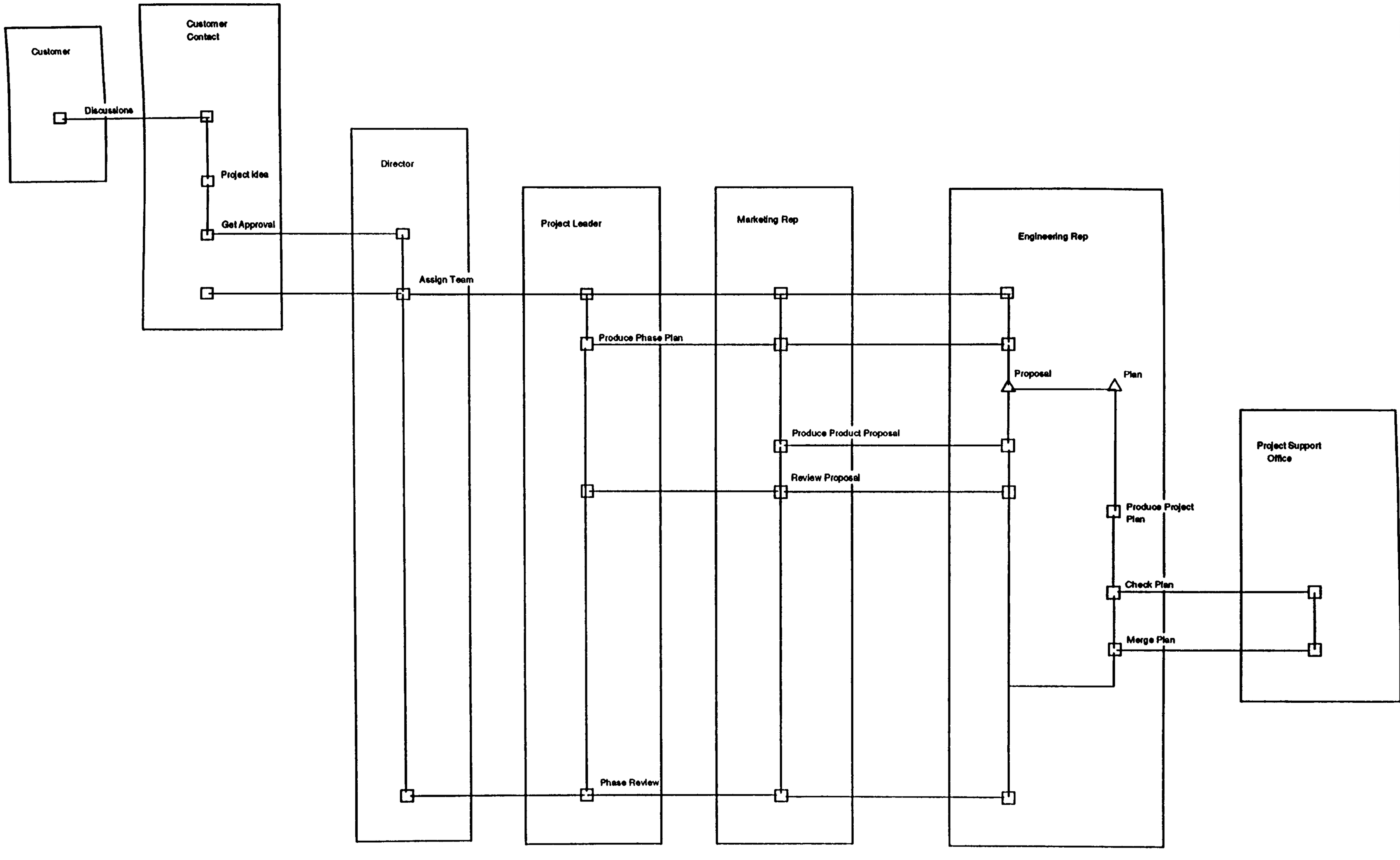
The comments of other business functions, however, are not implied by the process, and these functions are currently seen as external to launch. Hence the model explicitly shows a connection to them, with draft documents going to them and comments coming back.

PSO_Reports (store) =

{Report}

* Note that these may be reports about a plan which is still standalone. Merging may not have taken place *

'LE: Launch Process



Decision (Business goals)
.1

Decision (Process Area)
.3

Decision (Model_ orientation)
.2

Decide_ Notation
.4

Model_ Templates
.6

Model_ Procedures
.5

Model_based on_ Products
.7

Templates_ Model

Procedures_ Model

Products_ Model

Note_ Discrepancies _and_problems
.8

Note_ Discrepancies _and_problems
.9

Discrepancies _and_problems

Formulate_ Questions
.10

Questions

Insightsbases

Interview_ Actors
.11

View_of_Process

Model_ Actual_ Process
.12

Model_of_Process

Insights

Write_Report (Discrepancies, problems, etc)
.13

Report

Write_Report (Conclusions)
.15

Report

Validate_ Model
.14

Validated_ Model

Presentations _and_ Discussions
.16

Agreed_Report

Decide_upon _action (Future_ directions)
.17

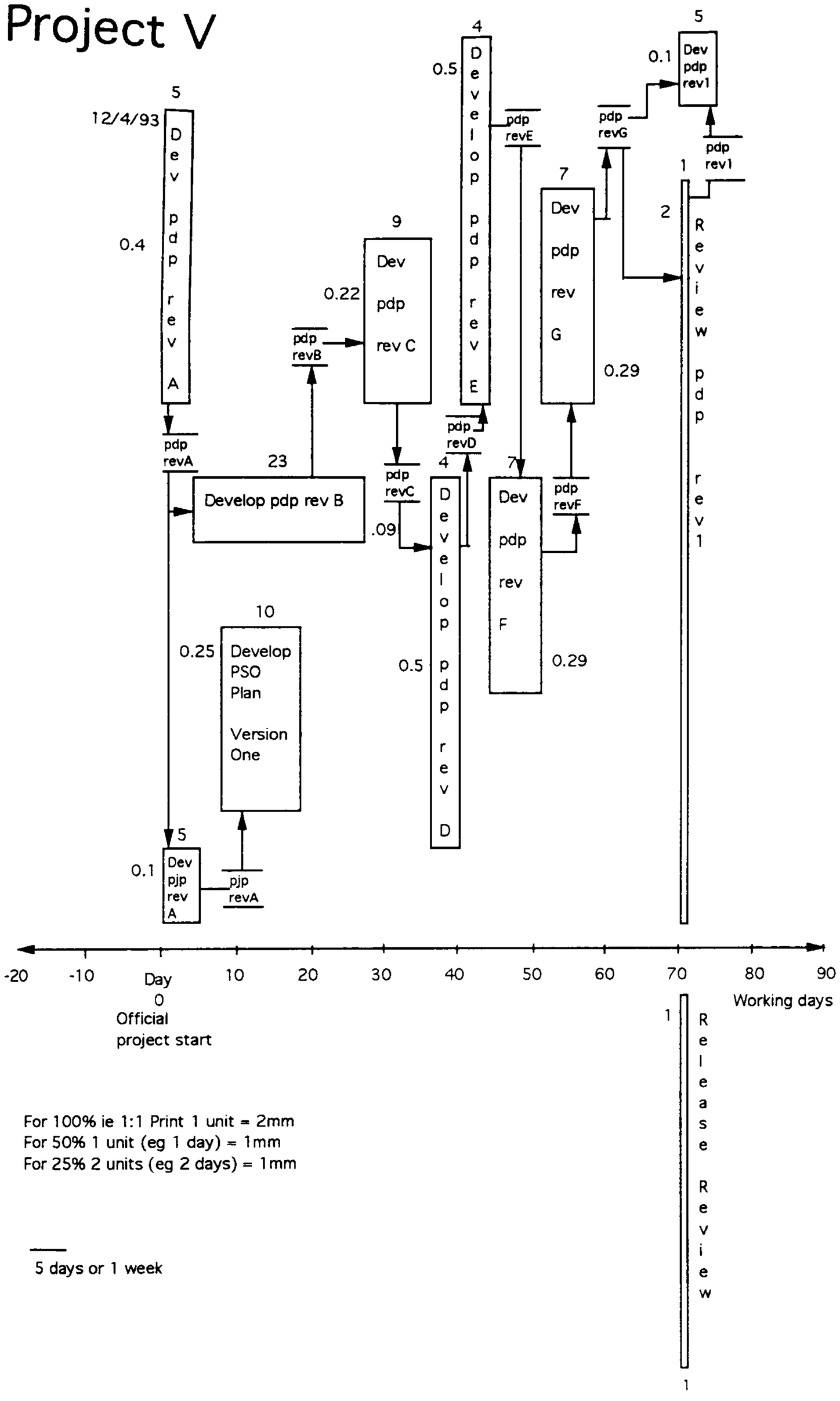
Process_Model

Propose_new _model/ process
.18

Appendix B: Models for the Five (Instance) Projects: V, W, X, Y and Z

Horizontal : Each unit = 1 day (actual time axis).
Vertical : Each unit = 0.01 people per day (only a scale).

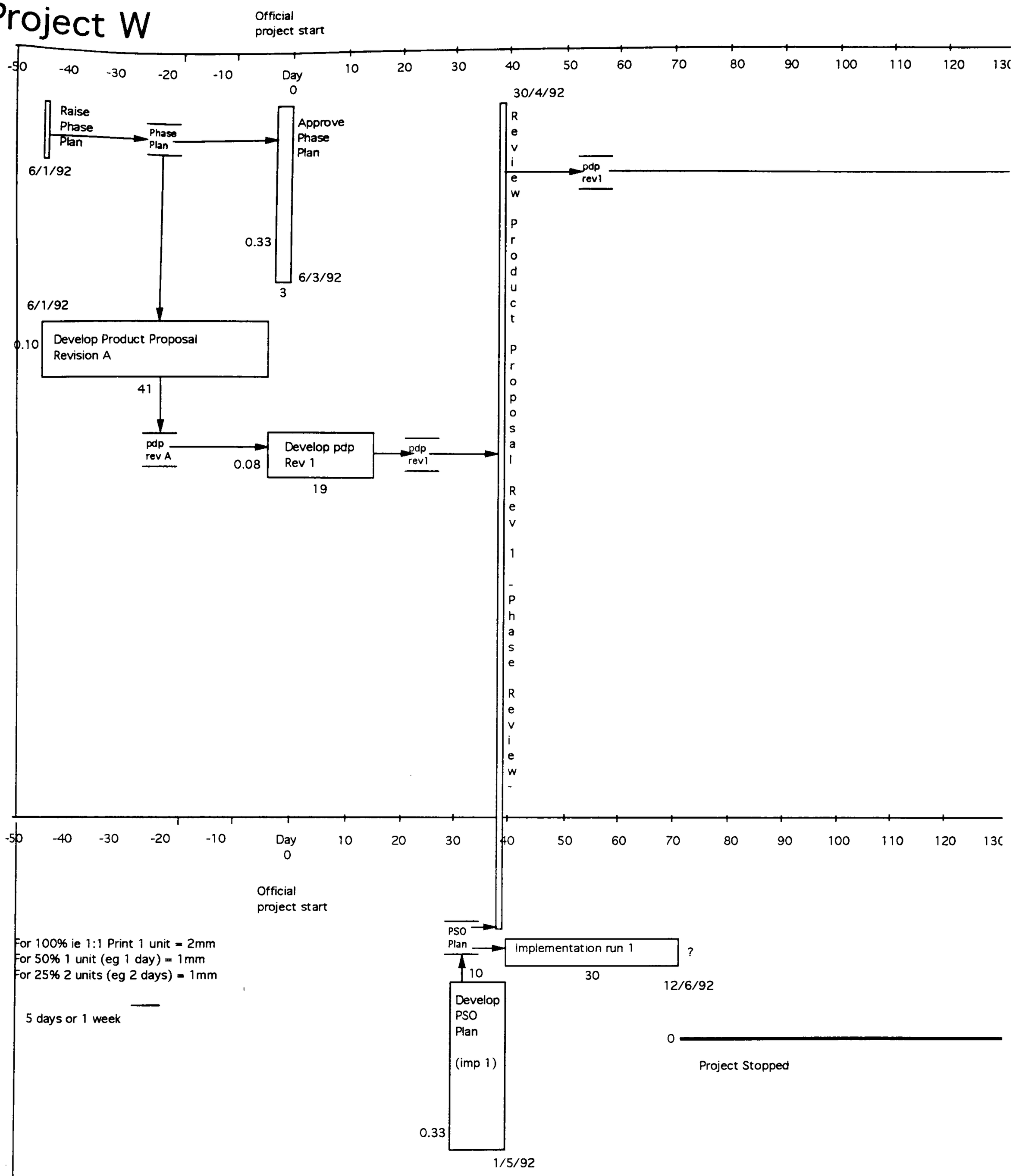
Project V

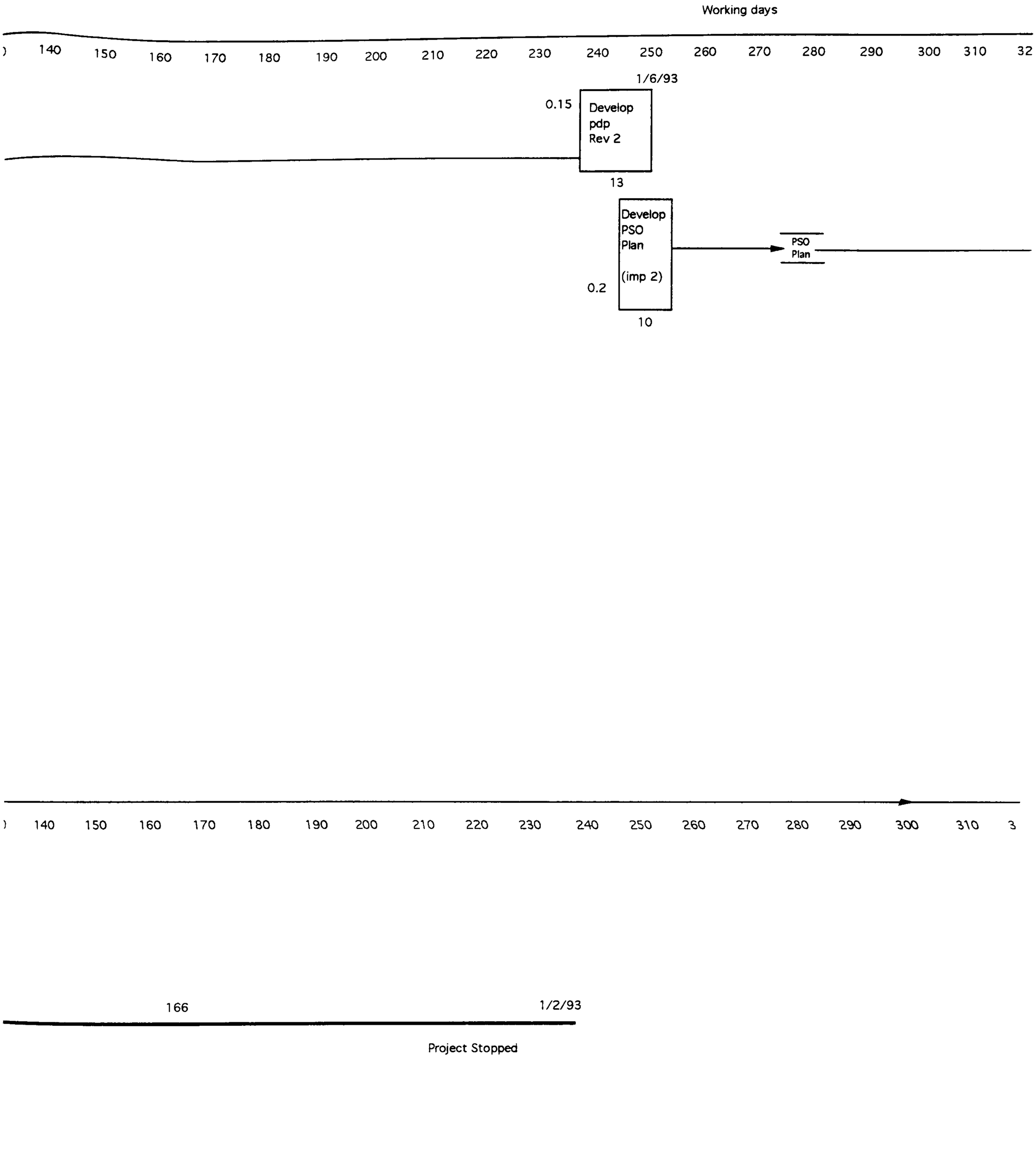


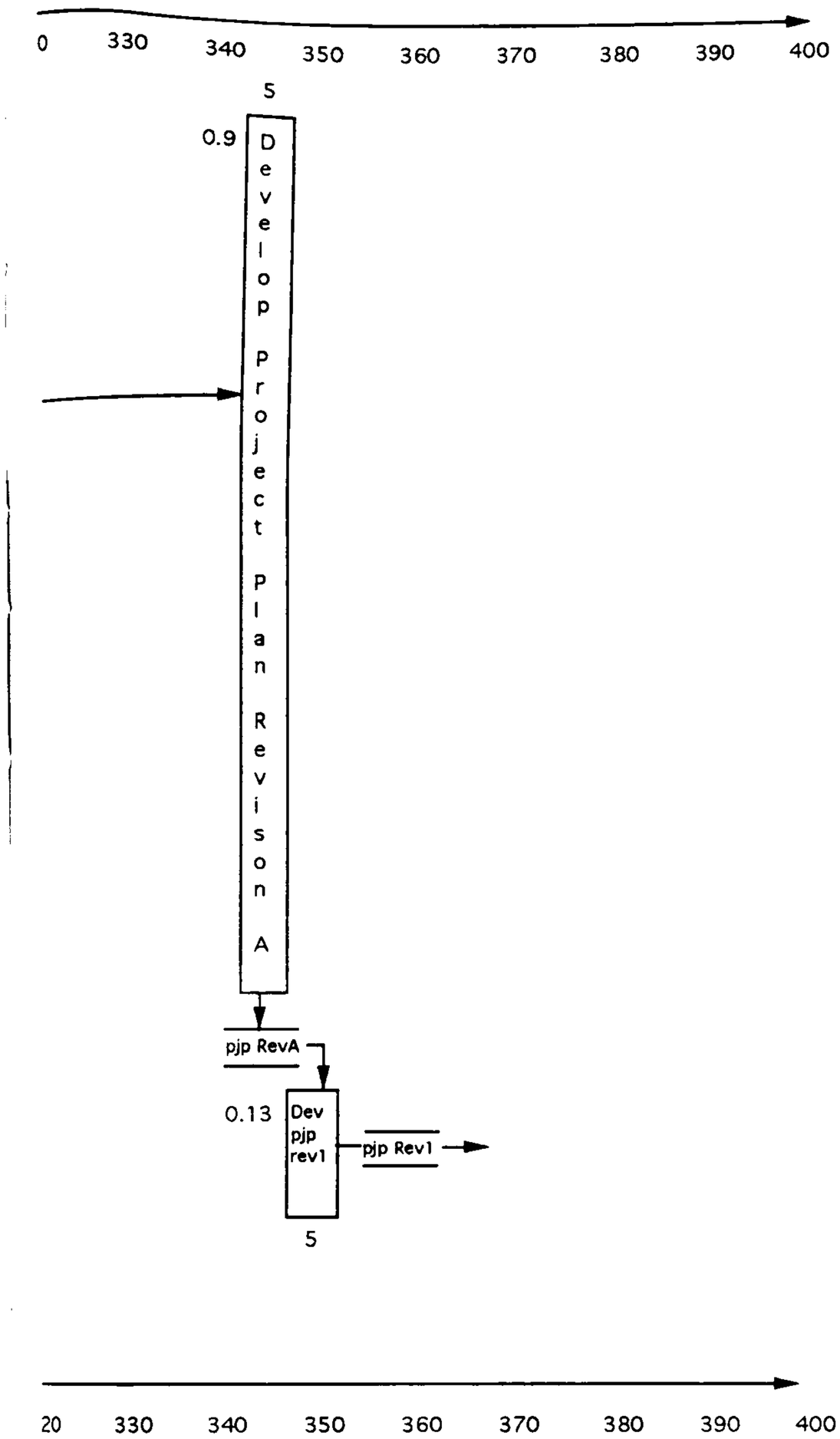
For 100% ie 1:1 Print 1 unit = 2mm
For 50% 1 unit (eg 1 day) = 1mm
For 25% 2 units (eg 2 days) = 1mm

5 days or 1 week

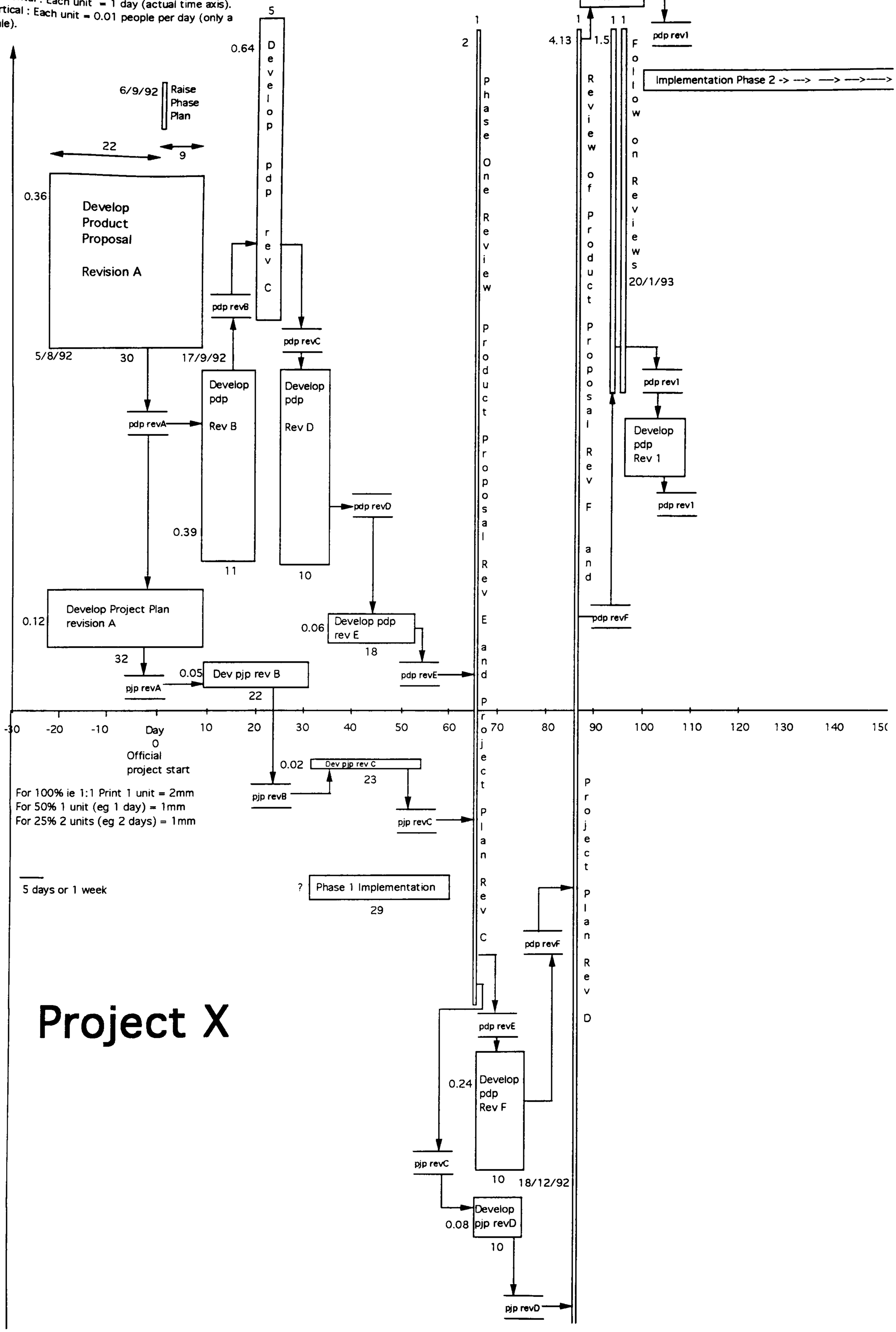
Project W

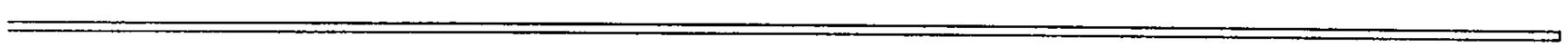

Official
project start
-50
-40
-30
-20
-10
Day
0
10
20
30
40
50
60
70
80
90
100
110
120
130
Raise
Phase
Plan
6/1/92
Phase
Plan
Approve
Phase
Plan
0.33
6/3/92
3
6/1/92
0.10
Develop Product Proposal
Revision A
41
pdp
rev A
0.08
Develop pdp
Rev 1
19
pdp
rev1
30/4/92
Review Product Proposal Rev 1 - Phase Review - pdp rev1
Official
project start
PSO
Plan
Implementation run 1
?
10
30
12/6/92
Develop
PSO
Plan
(imp 1)
0.33
1/5/92
0
Project Stopped
For 100% ie 1:1 Print 1 unit = 2mm
For 50% 1 unit (eg 1 day) = 1mm
For 25% 2 units (eg 2 days) = 1mm
5 days or 1 week

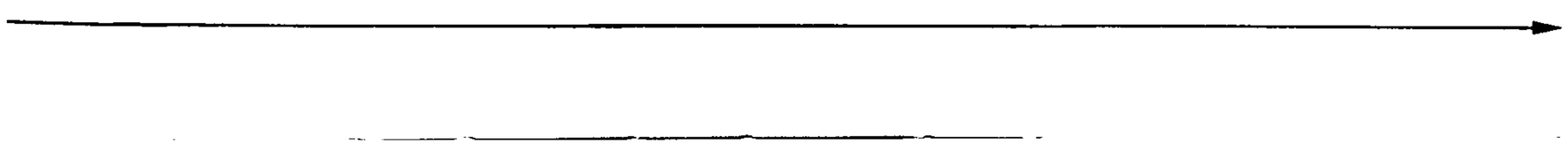

Working days
) 140 150 160 170 180 190 200 210 220 230 240 250 260 270 280 290 300 310 32
1/6/93
0.15
Develop
pdp
Rev 2
13
Develop
PSO
Plan
(imp 2)
0.2
10
PSO
Plan
) 140 150 160 170 180 190 200 210 220 230 240 250 260 270 280 290 300 310 3
166
1/2/93
Project Stopped


0
330
340
350
360
370
380
390
400
5
0.9
Develop project plan Revison A
pjp RevA
0.13
Dev pjp rev1
pjp Rev1
5
20
330
340
350
360
370
380
390
400

Project X

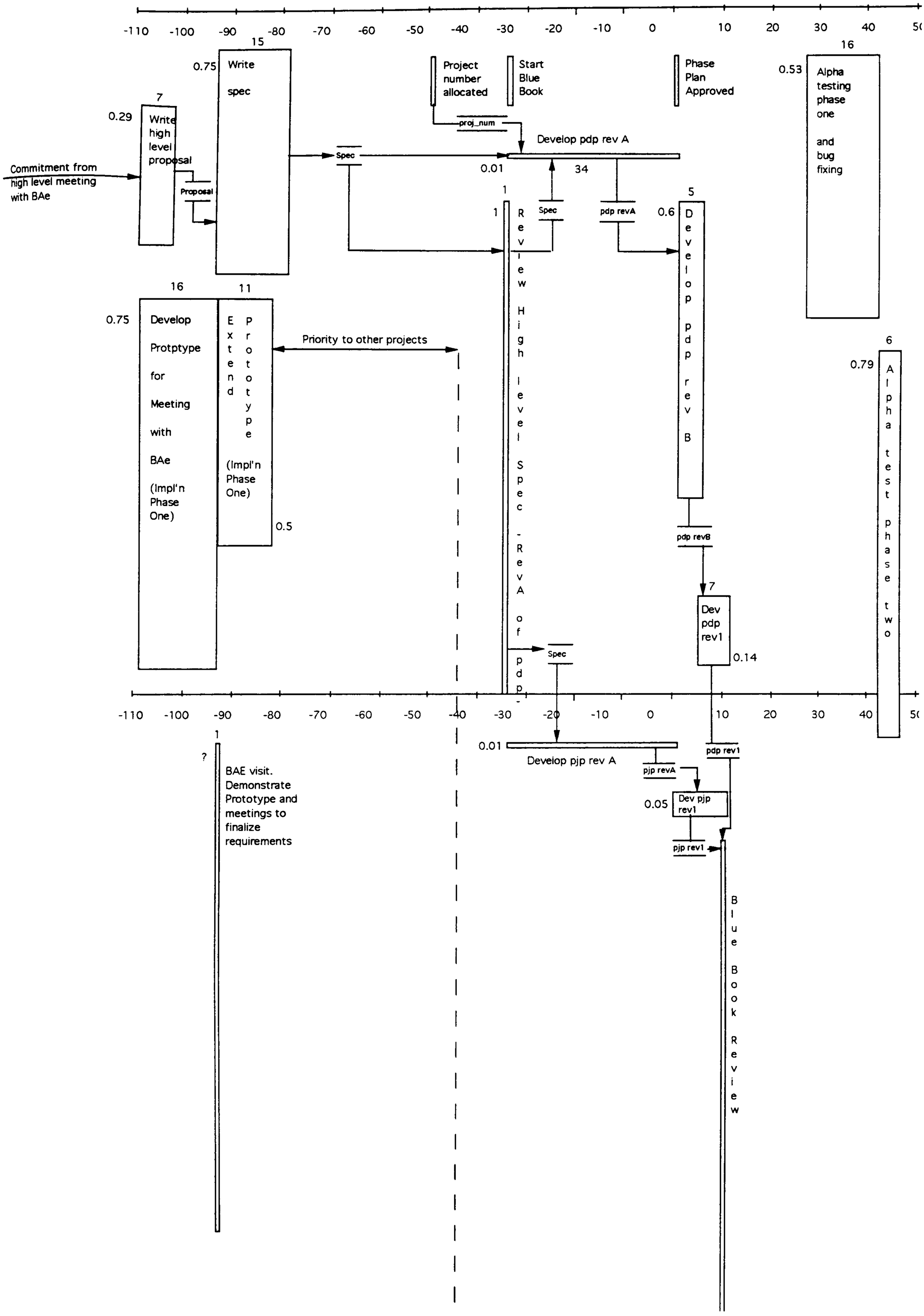

Horizontal : Each unit = 1 day (actual time axis).
Vertical : Each unit = 0.01 people per day (only a scale).
6/9/92
Raise Phase Plan
22
9
0.36
Develop Product Proposal
Revision A
5/8/92
30
17/9/92
0.64
Develop pdp rev C
5
pdp revB
pdp revC
Develop pdp Rev B
11
Develop pdp Rev D
10
0.39
pdp revA
pdp revD
0.12
Develop Project Plan revision A
32
0.06
Develop pdp rev E
18
0.05
Dev pjp rev B
22
pjp revA
pdp revE
-30
-20
-10
Day 0
Official project start
10
20
30
40
50
60
70
80
90
100
110
120
130
140
150
0.02
Dev pjp rev C
23
pjp revB
pjp revC
For 100% ie 1:1 Print 1 unit = 2mm
For 50% 1 unit (eg 1 day) = 1mm
For 25% 2 units (eg 2 days) = 1mm
5 days or 1 week
?
Phase 1 Implementation
29
1
2
Phase One Review
Product Proposal Rev E and
Project Plan Rev C
Dev pjp rev1
1
4.13
1 1
1.5
pdp rev1
Implementation Phase 2 -> --> --> --> -->
Review of product proposal Rev F and
Follow on Reviews
20/1/93
pdp rev1
Develop pdp Rev 1
pdp rev1
pdp revF
Project Plan Rev D
pdp revF
pdp revE
0.24
Develop pdp Rev F
pjp revC
10
18/12/92
Develop pjp revD
0.08
10
pjp revD

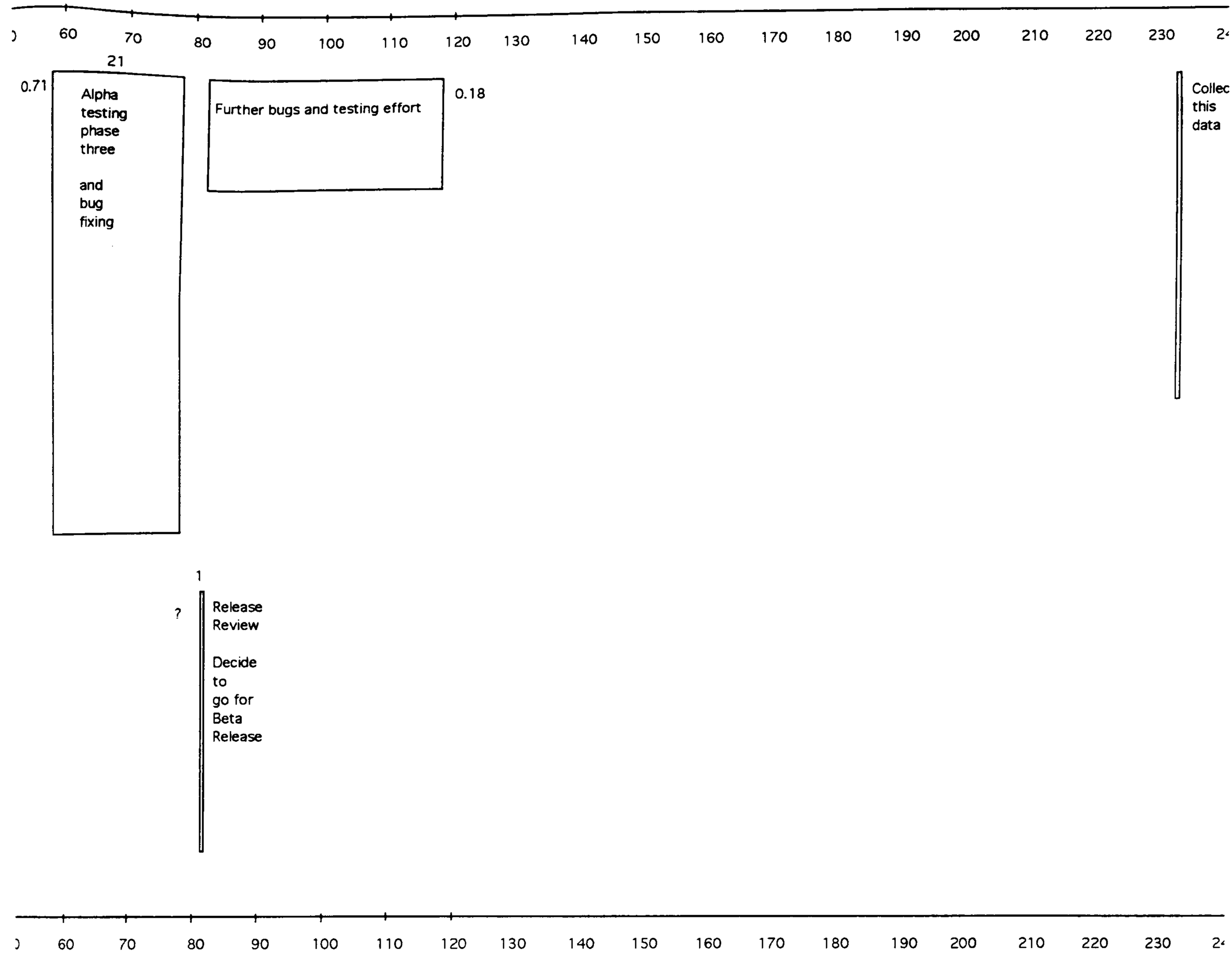


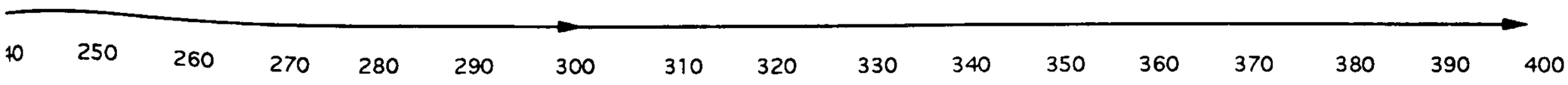



160
170
180
Working days

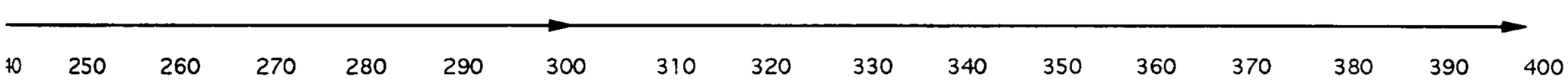
PROJECT Y

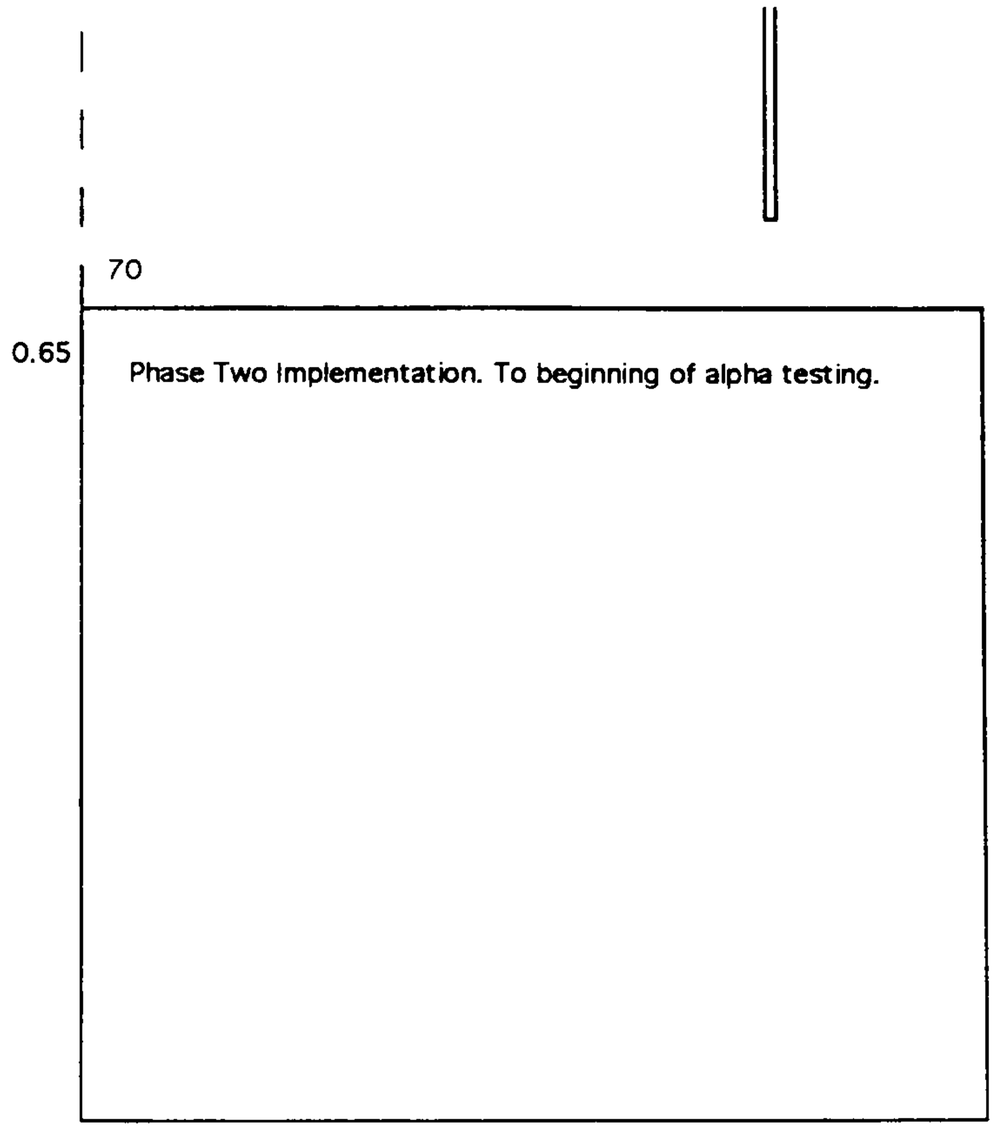

-110 -100 -90 -80 -70 -60 -50 -40 -30 -20 -10 0 10 20 30 40 5(
15
0.75
Write
spec
7
0.29
Write high level proposal
Commitment from high level meeting with BAe
Proposal
Project number allocated
Start Blue Book
Phase Plan Approved
proj_num
Develop pdp rev A
Spec
0.01
34
1
1
Review High level Spec - RevA of pdp
Spec
pdp revA
5
0.6
Develop pdp rev B
16
0.53
Alpha testing phase one and bug fixing
16
11
0.75
Develop Protptype for Meeting with BAe (Impl'n Phase One)
Extend
Prototype
(Impl'n Phase One)
0.5
Priority to other projects
6
0.79
Alpha test phase two
pdp revB
7
Dev pdp rev1
0.14
Spec
0.01
Develop pjp rev A
pdp rev1
pjp revA
0.05
Dev pjp rev1
pjp rev1
1
?
BAE visit. Demonstrate Prototype and meetings to finalize requirements
Blue Book Review


60
70
80
90
100
110
120
130
140
150
160
170
180
190
200
210
220
230
21
0.71
Alpha testing phase three
and bug fixing
Further bugs and testing effort
0.18
Collec this data
1
?
Release Review
Decide to go for Beta Release
60
70
80
90
100
110
120
130
140
150
160
170
180
190
200
210
220
230

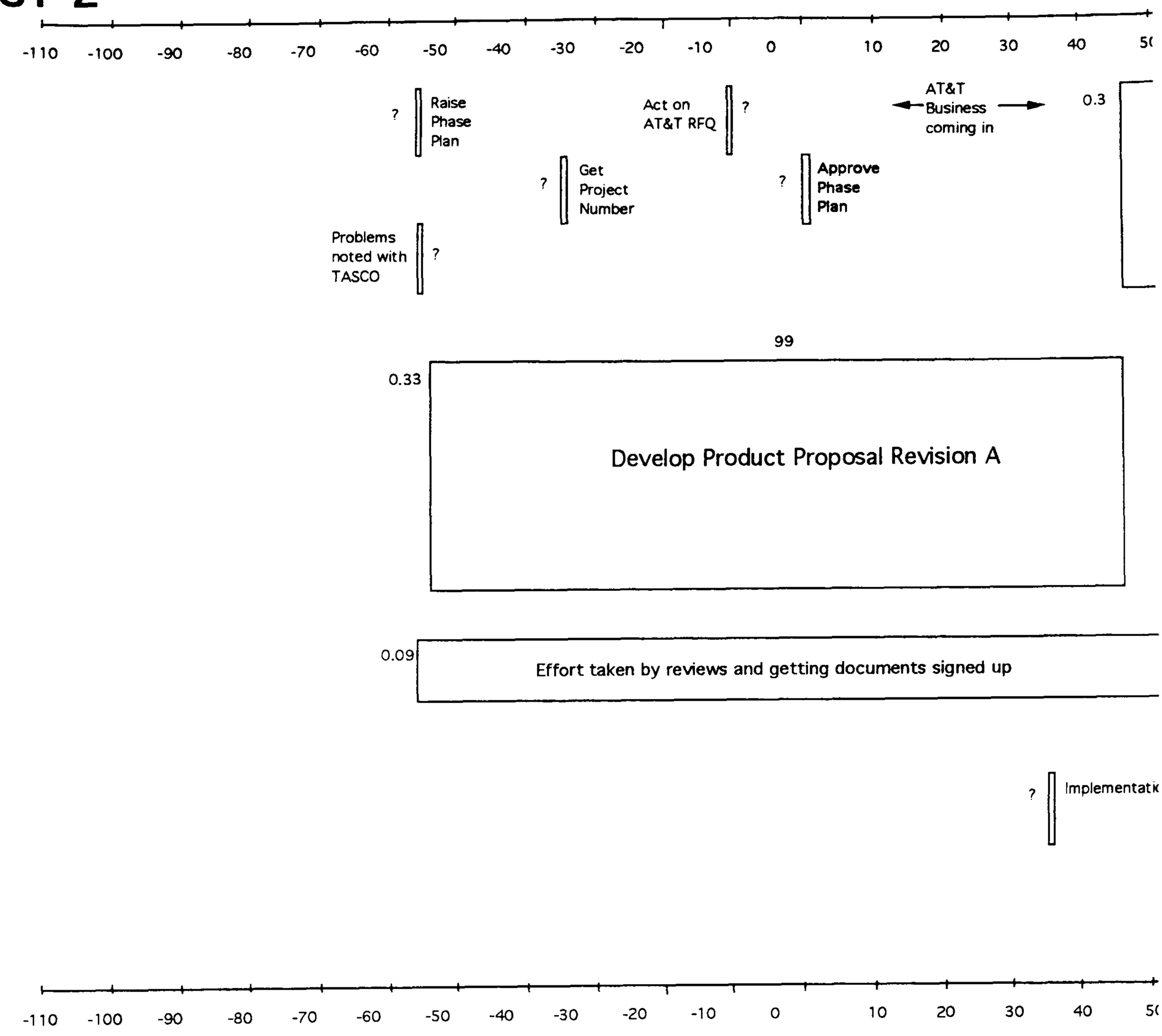

40
250
260
270
280
290
300
310
320
330
340
350
360
370
380
390
400

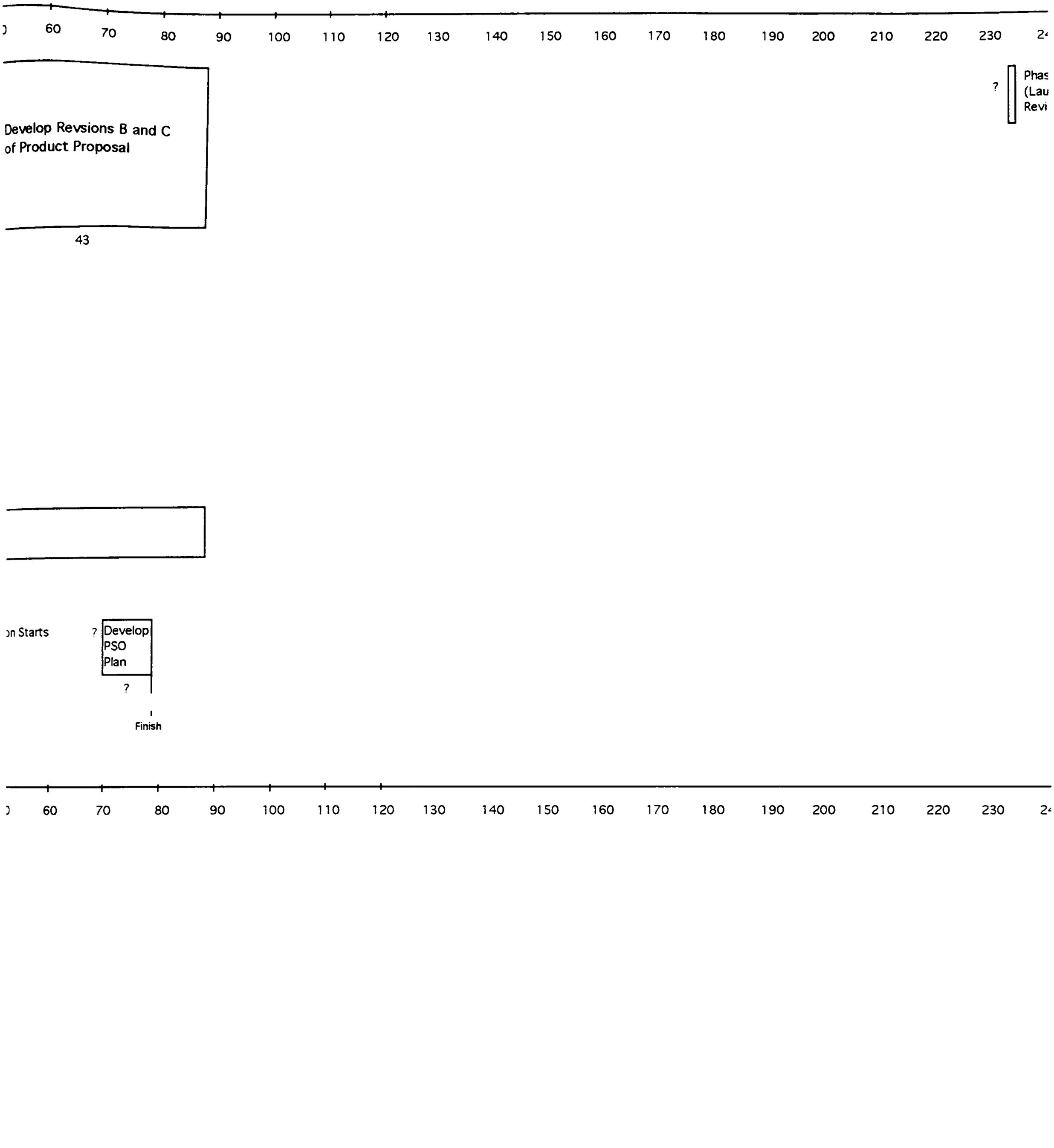
:ting

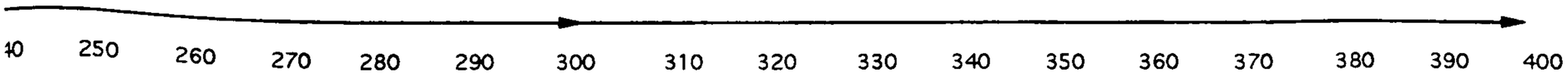

40
250
260
270
280
290
300
310
320
330
340
350
360
370
380
390
400


70
0.65
Phase Two Implementation. To beginning of alpha testing.

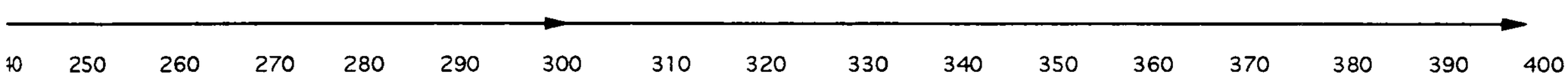
PROJECT Z


-110 -100 -90 -80 -70 -60 -50 -40 -30 -20 -10 0 10 20 30 40 5(
? Raise Phase Plan
? Get Project Number
Problems noted with TASCO ?
Act on AT&T RFQ ?
? Approve Phase Plan
AT&T Business coming in
0.3
99
0.33
Develop Product Proposal Revision A
0.09
Effort taken by reviews and getting documents signed up
? Implementatic


60
70
80
90
100
110
120
130
140
150
160
170
180
190
200
210
220
230
Develop Revsions B and C
of Product Proposal
43
Develop
PSO
Plan
Finish


40
250
260
270
280
290
300
310
320
330
340
350
360
370
380
390
400

se
inch)
ew


40
250
260
270
280
290
300
310
320
330
340
350
360
370
380
390
400