INTERFACING WITH THE SOFTWARE INDUSTRY - PART I -

THE ORGANIZATION OF CONTRACTED SOFTWARE

by

KARL REED

National Chairman,
Software Industry Committee,
Australian Computer Society

ABSTRACT

The questions of project specification, progress monitoring and testing and project organization are reviewed allowing for the influence upon the conventional wisdom of the presence of a software contractor. It is argued that many of the views on these matters presented in the literature do not apply under these circumstances.

In particular, development should be frozen or strictly controlled. Specifications should include resource statements and, project design should be such that another contractor should be able to take over.

Project organization should rest control in a manager who has both complete executive power, and responsibility. The client should not interfere.

Keywords and phrases: software contract, project organization, project specification, progress verification, Brooks' Law, separable design.

CR Categories: 2.2, 4.9

Authors Address:
c/o Computer Centre, Monash University,
Wellington Rd., Clayton, Victoria 3168, Australia.

INTERFACING WITH THE SOFTWARE INDUSTRY - PART I

THE ORGANIZATION OF CONTRACTED SOFTWARE

OVERVIEW

This paper is the part one of a series on 'Interfacing with the Software Industry'. The second part, also presented at this workshop, is called 'Some Contractual and Legal Questions Affecting Contracted Software' (Reed, 1978).

The two papers are self-contained and are intended as a first reading on their respective subjects, rather than the last word.

1. INTRODUCTION

Australian computer users will be forced to make more extensive use of the software industry as the expected short-fall in staff takes effect (see Ferranti and Smith). The relationships between software supplier and customer are similar to those between an engineering firm and its clients with the difference that there is not the same body experience, legal precedents and accepted standards which act to regulate the situation.

This paper deals with the technical and organizational aspects of project organization as applied to a software project being performed under contract. Much of the material on project organization is well known, however, the separation of responsibilities that may occur makes such projects qualitatively different from those undertaken by one organization.

The particular aspects of the relationship between a software company and it's client which have been chosen for consideration are:-

a) The Project Specification
b) Progress Monitoring and Testing, and
c) Project Organization

All of these are treated extensively in the various works referenced later, but only in the context of a project being undertaken by an EDP department for it's own users (so called 'In-house' development).

The view taken here is that the involvement of another party (the software contractor) materially alters the situation. The conventional wisdom then does not apply, as will be seen. More general questions of project management are left to authors such as Metzger (1973) and Weinberg (1971).

2. THE PROJECT SPECIFICATION

2.1 Producing The Specification

'Plan to throw one away; you will, anyhow' Brooks (1975 p. 16)

Brooks' law rings like the knell of doom in designers ears. The unmistakable implication of this is that system implementation should be a 'recursive development cycle' (Podalsky 1977), and that 'programmer's will find their debugged programs modified to account for changes in user requirements' (Swanson). Figure 1 shows Podalsky's view of the development process.

Recursive Development Cycle (Exhibit 2)

1. Feasibility Study - also 'impact' study; what changes in the basic functioning of the department should be (can be) via the system.
2. External Specification - first cut.
3. Internal specification - key factors - ease of change, promptness.
4. System construction - key factors - ease of change, promptness.
5. Implementation - training, begin wish book, plan for enhancement.
6. Familiarization period - settle in, continue wish book.
7. External specs for next version - requires priority setting with users.

Fig. 1 Podalsky's (1977) version of the development cycle.

This author finds such pronouncements frightening, and suggests that any client faced with a highly plastic implementation project for which he was paying on a time and materials basis would do also! There is some comfort in the fact that other authorities recommend that the specification be frozen prior to implementation. (Reid (1978), Evans & Assoc. (1975), Montgomery (1970) and Metzger (1973)).

However, a specification failure, that is, a situation where the system does not work or is unacceptable to the users even though it works can be classed as a software contract crash. Various authors, Podalsky (1977), Keen and Gerson (1977) and Marquet (1977) would agree that this can occur.

Neither client or contractor wants to be further experimental evidence of Brooks law, so steps should be taken to ensure that they are not while recognizing the potential truth of the proposition.

Client - Contractor Interaction And Recursive Design

It is common for clients to ask contractors to quote on a complete frozen specification. This tactic is adopted to enable a fixed price and delivery date to be set.

The evidence of the major writers is that this is poor practice, since the one that gets thrown away might be the one just paid for! An evolutionary implementation approach may be acceptable if one wishes to keep a large in-house EDP department busy but it would not be advisable to allow the specification to be altered continuously during a contract, since, as already mentioned, costs would become uncontrollable.

It would seem therefore that the gods will be placated if there is a design iteration to produce a frozen specification, and it would also seem good sense to involve the contractor. The original specification can then be discarded, having served its purpose in helping to select the supplier. It will also serve as a guide in the further stages of the process.

Such an approach will be rejected by those clients who have already 'iterated' to their specification but it is suggested that the interests of both parties might be served by attempting to 'shake' the specification before implementation begins. The contractors are surely just the people to assist in this process. Figure 2 shows appropriate modifications to Podalsky's cycle.

A Recursive Development Cycle For Contract Software

Phase:	Performed By:
1. Feasibility Study	Client and User
2. External Specification	Client and User
3. Supplier Selection	Client
4. Reassess Feasibility	Contractor, client, user
5. Confirm External Specification	Contractor, client, user
6. Internal Specification and basic design	Contractor, client, user
7. Implementation	Contractor
8. Familiarization	Contractor
9. Commence new contract if further development needed	Contractor

Fig. 2 Development cycle for contracted software (Reed).

Producing The 'Final' Specification

The process of producing the final specification must include the users in the client department. Modern writers such as Jones (1977) recommend involving the users in higher level 'walk through's' to enable them to see what is actually being proposed. So do Keen and Gerson (1977).

There may be some point, if the project is large enough to justify the effort, in creating a mock-up system which the users pretend to use, (Refer Constantine and Yourdon (1975) pp. 513-514), so that they can set some feel for the value which the system might have.

Participation in the conceptual design, at the level at which reports, processes and basic collections of data are being fixed will give the users confidence in their capacity to predict their requirements, and confidence in the system itself. The supplier will also have a chance to interact with the end-user.

Above all, however, the specification should be that of a clean architectural entity, as stressed by Brooks, (1975 pp. 42-43). In other words, it should contain all relevant information and be readily understood. This is best achieved if it is written by one person. (Brooks, 1975).

2.2 Factors To Be Included In The Specification

The quality of the specification is critical. It must be precise and show the information content of terminal dialogues, reports, data layouts and system function. The basic structure of a specification is described
in such works as Metzger (1973) and the impact of standards upon the specification are covered in
Reed (1978).

The relationship between the system and it's hardware environment are equally important and tend to receive less attention in the literature, which is rather short in descriptions of usable quantitative design techniques in any case. Factors which must be included are terminal response times, memory usage, external storage requirements, unit record peripheral utilization and frequency of run. These things are usually not too difficult to estimate once the system function is decided.

Designers experience greater difficulty in estimating the CPU utilization per transaction, communication line occupancy, the number of disc accesses per transaction and other resource requirements which actually determine the performance. Factors such as virtual memory paging rate attract even less attention.

The items listed above determine both the actual system performance and it's impact upon the machine upon which they run. Whether or not the existing machine will have the capacity to accept further systems is determined by these factors. There is also a case for specifying the existing system load, in terms of available CPU time, channel and disc accesses. It is just possible that the system will not run on the proposed installation under normal operating circumstances even if the configuration will support it.

This author would argue that an attempt should be made to put upper limits on the resources available to the completed system. A contractor responding to the tender is then left in no doubt about the real difficulty in achieving the specified end user performance.

Other obvious factors to be included are languages to be used, standards of documentation and acceptance testing.

Table I

Information On Resources Utilization To Be Included In Specification

Resources required by:

Proposed System.
Systems with which it will run concurrently.
Installation slack, for future development.

Table II

Resources To Be Specified For Categories In Table I

Resource	Information Specified
CPU	Utilization per transaction and overhead
Memory	Profile of requirements with transaction types etc.
Unit Record	Line printer utilization, card reader etc.
Communications	Line occupancies, transmission speeds
Virtual Memory	Disc access rates (requests-sec)
Rotating Storage	Pack mounts, access rates
Magnet Tape	Tape mounts, access rates

2.3 Resolving Errors In The Specification

Correcting an inconsistency in the specification will involve a volume of work which is an increasing function of the percentge of work completed at the point of discovery.

The problem of deciding the responsibility for such changes is covered elsewhere (see Reed 1978) and in this paper.

2.4 Separable Design And Incremental Implementation

Separable Design

It has been suggested by Symonds (1977) that it is possible to design and implement a system in such a way that another project team can take over if the first is dismissed. This, he argues protects clients against the effect of a contractor defaulting during the project since moneys outstanding in progress payments may not compensate him for the delays which will ensue.

The implications of such an approach need careful analysis.

A close look at Brooks' laws, ('adding manpower to a late software project makes it later.' Brooks (1975) page 25) taken by Gordon and Lamb (1977) suggests that the cumulative losses might be unbearable. What may come to be known as Marquet's Law ('the maximum recoverable value of incomplete, untested production is zero.' Marquet (1977) page 48) is even more pessimistic. Reid's analysis of the Western Australian Library project (Reid, 1978) contains an example of what can happen.

It is clear then that Marquet's Law must be violated and Brook's Law avoided if separable design is to work.

Marquet's Law can only be nullified if the basic design is extremely modular and the interface testing of very high quality. Interface testing must be carried out at frequent intervals, and the test data and its results kept on machine readable media for all milestones in the project. Testing would also need to be extremely well documented so that the new project team could readily reproduce the earlier tests when necessary. All of this would mean that test harness software may be needed.

Brook's Law can only be circumvented if the total project information structure is transparent, and the total project experience can somehow be instantaneously absorbed by the new project team. This is impossible, so the best which can be expected is that the specification is clear, that perhaps some design and progress meetings are well documented or taped and that the existing work is well documented as recommended by Brooks (1975). The client just might minimise his losses under these circumstances.

The question which must now be asked is 'But isn't this how it should be done anyway?' The answer is yes. It is not difficult to find recommendations along these lines in the literature (for example Poole (1973), Yourdon (1975) chapter 6, Herman (1975), Hetzel (1973), Martin (1967) to name just a few).

This writer suspects, however, that clients who insist upon a system being implemented in a 'separable' fashion will find themselves being quoted a higher price for the contract, which of course is an interesting commentary on the general state of systems implementation.

Incremental Implementation

The topic deserves some comment in view of the various authors referenced earlier, (section 2.1).

This writer's view is that the contract should also be incremental, so that the client is not committed beyond the current increment, as pointed out by Metzger (1973) page 9.

3. PROGRESS MONITORING, TESTING AND VALIDATION

3.1 Implementation Milestones And System Structure

Statements referring to percentage completion are meaningless unless there is some means of verifying that the work has been actually done. The '90% complete' scenario in which software remains 90% complete indefinitely is well known (see Ingrassa, 1978).

Attempts therefore to monitor progress become related to system structure. A system design which is incremental, i.e. involves the construction of a primitive system which will evolve through defined phases until the final product, perhaps consisting of several fairly separate but complete systems exists, will be easy to monitor. Various milestones can be equated to the sub-system level and tests designed to verify that they function as intended.

This may not be sufficient if the various sections are sufficiently large. The client may then need to specify that the existence (and correct operation) of system fragments is to be demonstrated to verify progress. In other words, some of the requirements for separable design may intrude into the area of progress monitoring.

Some suppliers will again argue that the production of intermediate test results in a form suitable for analysis by the client constitutes significant additional work, and may increase their quotation accordingly. It may even be argued that the implementation will be significantly delayed, made less efficient etc. by such requirements, placing the prospective purchaser in a difficult position.

The client should also note that a significant amount of work may be needed within his own organization to specify these milestones in a sensible fashion, and to verify that they are met.

3.2 How Much Progress Monitoring Is Necessary?

The amount and extent of progress monitoring will depend upon the system being built and the constructor. It may well be that a contractor who has completed several similar systems on the same hardware can claim that progress does not need detailed monitoring. Careful examination of both the systems mentioned as evidence of competence, and of the design team proposed is then in order.

The last point is important since experience is the property of individuals rather than a company. One would expect that the implementation team should contain appropriate numbers of those staff involved in the previous projects, and the client would be well advised to check this out.

The value of very high levels of experience and competence cannot be underestimated, since they may mean that much of the previous discussions are not applicable, and that implementation times, if not costs, are greatly reduced. This should hearten those readers who are beginning to feel that the whole deal is not worth the candle.

A note of caution must be sounded. One must be extremely confident before basic safeguards such as progress monitoring are relaxed.

3.3 A Question For Mutual Agreement

The exact nature of milestones and progress monitoring is, as that of progress payments, a question for negotiation, since, as has been pointed out, these must be related to the system structure.

4. PROJECT ORGANIZATION

4.1 Chains Of Command

The primary mechanism of project control, from the clients point of view, should be through the progress monitoring discussed in section 3. Responsibility for the success of the contract should lie with the project manager, and the client should not interfere with him unless some catastrophy is impending. This applies even if the project manager is provided by the client.

It is a good idea, if the project is large enough and uses the clients installation for development, for the client to provide a senior staff member to assist the project team in obtaining the physical resources that it needs. This 'expediator' should be under the control of the project manager, and should have enough authority within the client organization to ensure that the project is not delayed because of physical resource shortages.

Weinberg (1971) suggests that a devil's advocate is also necessary. This person's job is to attend project meetings and to explore all possible modes of failure and miss-reporting. Whether or not the clients representative should adopt such a role is debatable since the project team may close its ranks against him, with the result that the reporting accuracy may drop.

This author's view is that the chain of command should be clear, and as simple as possible. As already mentioned elsewhere, this should hold independent of the actual project team composition. Customers' staff should attempt to resolve problems within the project team and only involve their own project supervision when they are unable to do this, or if a serious problem arises.

4.2 Problems Due To Joint Staffing

The idea of both supplier and customer staffing a project has very significant appeal. It tends to keep the contract price down and ensures that the customer has staff capable of maintaining and extending the system. It may also be considered desirable to have staff directly involved in the project who can assist in monitoring it.

This writer's view on the latter point should be clear by now, so no comment will be made. Other problems which may arise will be addressed instead.

John Marquet presented an interesting paper in 1977 which related the software industry to the animated cartoon industry (Marquet 1977, pages 45-48).

Marquet's comments upon division of responsibility bare direct quotation. He argues that a software company is like a cartoon producer.

Table III

Identity Equivalents Between Cartoon and Software Projects

Cartoon Identity	Software Identity
Producer	Application Manager
Art Director	Programming Leader
Animator	Application Programmer
Background Artist	System Programmer
Layout	Vendor-supplied Software
Scenarist	System Analyst
Dubbed Voices	User Training
Editor	System Integration
Camera	Development Computer
Theatre	Production Computer
Theatre Owner	User Management
Audience	User Personnel

'Suppose some corporate tigers wander into one of our cartoon companies to buy something; it could be a feature, a training cartoon or an animated commercial. The cartoon company front men try to explain about Art Direction, Production Values, etc., as well as why Bugs Bunny is not quite right or otherwise unavailable for the project under discussion. The cultural shock is often considerable, and leads to a primary project pitfall.

In our experience cartoon/software buyers frequently fail to recognise the need for all the identities listed in the above correspondence table. They may unreasonably assume, for example, that the cartoonists have access to block time on a camera of the right brand (while the cartoonists assume nobody could overlook such a detail). They may expect the Art Director to go about telling future audiences how good it will be, or they may expect the cartoon company to use in-house animators (many of whom are actually tracers).

The primary pitfalls of cartoon/software production stem from "loss" of one or more identities from the project environment. The result is a dismembered project. For example, many a computer system has been developed without analogous dubbed voices; the users are subsequently unable to tell the funny parts from the sad parts.

A cartoon/software buy can be successful only if someone is designated to look after each of the identities listed in the correspondence table shown above. Individuals can of course cover more than one item, and occasionally an item can be under the control of a committee. Some general rules probably apply; e.g. the cartoon/software company should never constitute its own audience, the Art Director should not be a committee drawn from vendor and client sources, but there are exceptions to these rules. There are also degrees of unhappiness for the project, depending on who or what was left out. It is not unusual for the following pitfall variants to occur:

Actual layouts (vendor supplied software) are found to be inadequate when animation commences.

Animators and/or the Art Director appear late in the project, and fight thereafter over "production values".

Audience reaction is not measured before or during the project.

So far, I have tried to illustrate the point that a software project team is a consortium of talents (regardless of project size). The project is an aggregate of people and facilities; the provision of the aggregate environment is a necessary condition for a successful software buy/sell transaction, but it is not sufficient. If the circumstances described are pitfalls (bits of the project falling down holes) we had best consider the consequent traps (nasty things to tread on as we cross the project terrain).'

Marquet's warning is against 'role blurring' and 'missing identities'.

The author has participated in a jointly staffed project which was rather successful. One of its salient features was the clarity of the roles, and the chain of command, despite the presence of senior customer personnel. The supplier took technical responsibility, and all levels of the project were responsible to the supplier's project manager, who reported periodically to a committee consisting of the clients senior manager responsible for the project (not a participant) and the client's senior man present on the project. It was known and accepted that this clients senior man present made reports directly to his management.

'Role definition' within the project was extremely clear, and everyone knew where they stood.

It would be too much to ascribe the project's success to this factor alone, but it certainly helped.

Extended arguments will arise due to the kind of interference which occurs when the client's staff are expected to be technical watch dogs reporting problems to their own management instead of the project leader. The effect will be delays and some loss of control, both technical and managerial, by the supplier who could then quite rightly feel that penalty clauses should be waived and delivery dates altered.

The client would, one expects, be at a severe disadvantage before an arbitrator if it could be shown that such a situation had occurred on a delayed project. Arguments about relative responsibilities would become confused since it may no longer be clear who had executive power on the project.

It should be noted that no supplier should be expected to accept liability without responsibility and executive power.

4.3 Change Control

The reader will by now have gained (correctly) the impression that this writers view is that little or no change should be made to the specification, and to the design once it is complete. All change will cost something, and the cost will tend to increase as the project progesses. Some change is however unavoidable, and it must therefore be controlled.

Two types of changes can be identified; specification changes, and design and implementation changes. The first refers to what shall be done, the second to how.

One satisfactory technique for controlling changes is described by Metzger (1973) page 89-91. A very formal structure is suggested. Changes are to be notified on a 'Change Form' and investigated to classify them into two types. Those which effect the basic documentation or would have a negative cost impact and those which have little effect and do not have negative cost impact.

A committee would consider all changes, and decide whether or not they will be accepted. Someone must also decide who accepts the negative cost impacts as well, although changes to the specification which benefit the client and are accepted by him should be funded by him. The implications of this are that either party should have the right to vetoe changes which they must fund provided they do not effect the projects successes.

This approach should also be used to control the correction of design errors - with financial responsibility going to those who provide the design.

5. CONCLUDING REMARKS

The last word hast not been said on this matter, and there is a sense in which the industry must wait for legal precedents to be established before some questions are resolved. Good practice in project organization will eventually be determined by the ease with which parties can prove liability in disputes rather than by practical demands. The authors view is that the two needs converge (see Reed 1978) and that it should, as a result, be possible to make recommendations on this matter which could have the imprimatur of a standard. Such action would mean that much unnecessary pain and cost will be avoided on both sides, and that the image of the industry increased.

6. ACKNOWLEDGEMENTS

Acknowledgements are due to D. Butchart, J. Marquet, J. Whittle and Dr. C. J. Bellamy for assistance in clarifying these ideas.

These papers were typed and set by ATL Datatronics Ltd. of Melbourne.

References

Brooks, F. P. (1975) 'The Mythical Man-Month' Essays on Software Engineering Addison-Wesley 1975

Brown, J. R. et al (1973) 'Automated Software Quality Assurance' in Hetzel (Ed) 'Program Tested Methods' Prentice Hall 1973 pp 181-203

Evans, R. W. & Assoc. (1975) 'Developing EDP Projects Successfully' EDP In-Depth Reports R. W. Evans & Assoc. Ltd. (Canada) Vol. 4, No. 11 July 1975.

Gordon, R. L. and Lamb, J. C. (1977) 'A Close Look at Brooks' Law' Datamation June 1977 pp 81-86

Herman, (1975) 'Data Flow Analysis in Program Testing' MSc Thesis, Monash University 1975 (Dept. Computer Science)

Hetzel, W. C. (1973) 'Principles of Program Testing' in Hetzel, W. C. (Ed) 'Program Test Methods' Prentice-Hall 1973.

Ingrassa, F. S. (1978) 'Combatting the 90% Complete Syndrome' Datamation Jan. 1978 pp 171-176

Jones, N. M. (1976) 'HIPO for Developing Specifications' Datamation March 1976 pp112-125

Keen, P. G. W. and Geerson, E. M. 'The Politics of Software Systems Design' Datamation November 1977 pp81-84

Marquet, J. (1977) 'Traps and Pitfalls in Software Dealing', in 'Soft-where, Soft-why, Soft-how' Proceedings of the ACS-SIC Seminar June 1st 1977 pp 43-50

Martin, J. (1967) 'Design of Real-Time Computer Systems' Prentice-Hall 1967 (chapter 37, 'System Testing')

Metzger, P. W. (1973) 'Managing A Programming Project' Prentice-Hall 1973.

Montgomery, A. Y. (1970) 'Aspects of Data Processing Management' ACJ Vol. 2, No. 2. May 1970 pp 71-88

Podalsky, J. L. (1977) 'Horace Builds A Cycle' Datamation, November 1977 pp 162-168

Poole, P. C. (1973) 'Testing and Debugging' in 'Advanced Course on Software Engineering' Lecture Notes in Economic and Mathematical Systems Vol. 81 pp 278-317 Springer-Verlag

Reed, K. (1978) 'Interfacing With the Software Industry - Part 2 - Some Contractual and Legal Questions Affecting Contracted Software' Proceedings of the Eighth Australian Computer Conference Canberra, 1978.

Reid, T. A. (1978) 'The Trials and Tribulations of an On-Line Computer Project' Australian Computer Bulletin Feb 1978 Vol. 2, No. 1 pp 6-14

Smith, B. W. and de Ferranti, B. Z. 'Computers and the Future of Education' and 'The Present and Future Use of Computers in Australia and Employment Implications'

Swanson, E. B. (1976) 'Computer Application System Development: Some Implications for Programming Practice' Data Management Vol. 4 No. 5 May 1976

Symonds, R. (1977) 'Development of Large-Scale Commercial Systems Without In-house Programmers' in 'Soft-where, Soft-why, Soft-how' Proceedings of the ACS-SIC seminar June 1st 1977 pp 43-50

Weinberg, G. M. (1971) 'The Psychology of Computer Programming' Van Nostrand, Reinhardt & Co. 1971.

Yourdon, E. (1975) 'Techniques of Program Structure and Design' Prentice-Hall 1975

Yourdon, E. and Constantine, L. L. (1975) 'Structured Design' Yourdon Inc. (1975) 1133 Avenue of the Americas New York, N.Y., USA