Project Management of Batch Control Projects
- How to avoid the Pitfalls -

Dr. Maurice J. Wilkins
Director of Process Automation and Control Systems
Millennium Specialty Chemicals
Jacksonville, USA
Phone: 904 910 2982       Fax: 904 924 2788
E-mail: m-wilkins@attbi.com

KEYWORDS
Software, Batch Control, Project Management, S88

ABSTRACT

Batch control projects have tended to be software intensive and often overrun substantially. In most cases this is due to a lack of foresight and planning in the early stages. The “S88 Era” has tried to alleviate this situation by providing a structured methodology – but again the flexibility of current control systems and a lack of planning can produce the same effect as having no structure at all. This paper aims to highlight some potential problems and pitfalls and how they may be contained without adversely affecting the outcome of the project.

INTRODUCTION

I was once having a discussion with a runner prior to the London Marathon, about the enormous task ahead of us. My fellow runner said that he was mentally breaking down the distance of 26.2 miles as two tens and a six. A seasoned runner spoke up from behind us and said, “if I were you mate I would consider it as two tens and a fifteen.” We both looked at each other in shocked disbelief. However, the experienced guy turned out to be speaking the truth because, as we discovered, due to bad planning at the beginning of the race, we hit “the wall” at about twenty miles and the last six miles seemed like at least fifteen. So, why am I telling this story?

The fact is, that all too often we do not assess the risks or plan well enough in advance on projects in general, but especially batch projects. I have worked on many batch projects over the past 20 years and want to use this paper to share my experiences. I will also discuss ways I have learned of planning for the pitfalls and trying to avoid them.
PROJECT FUNDAMENTALS

First – let’s go back to basics…

A project scope consists of a set of inter-dependent activities which must be performed over a given time period at a given cost. In an ideal world, the cost, time period and deliverables are indicated in an estimate developed prior to the beginning of the project. The accuracy of the estimate depends on several things, such as:

- Is the project a repeat of a previous project?
- Is all of the information available from the end user?
- Is it new technology?
- Is it “grass roots?”
- Do standards, procedures and methods already exist?
- Has a process model been developed?

and many others.

The project manager should participate in the initial estimate to ensure that it is as accurate as possible, including an assessment of the risks to the cost and schedule. He has to make sure, by the effective deployment of resources, that the project costs and schedule are maintained on plan, whilst being able to supply the contracted deliverables.

The project manager is responsible for recognizing changes to scope and being able to communicate cost and schedule impact to the end user. He must then be able to obtain a decision (and sign off) from the end user, before incorporating the changes into the project. Changes to scope are one of the biggest items which affect the schedule and costing of batch projects – especially “creeping changes”, those small changes which “are too small to warrant a change control procedure”. Hence, effective change control is an important factor in the project management of these projects. Change control will be discussed later in more detail.
PROJECT PHASES

The phases of a typical project, which are shown below, need to address 3 fundamental items, What will be delivered, how it will be done and when it will be done.

<table>
<thead>
<tr>
<th>Project Phase</th>
<th>Requirements</th>
<th>How/When</th>
<th>Implementation and Test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>User Requirement</td>
<td>Concepts Definition</td>
<td>Detailed Design</td>
</tr>
<tr>
<td>Deliverable</td>
<td>Fixed</td>
<td>Fixed or Cost Plus</td>
<td>Fixed or Cost Plus</td>
</tr>
<tr>
<td>Phase Cost basis</td>
<td>Fixed</td>
<td>Fixed or Cost Plus</td>
<td>Fixed or Cost Plus</td>
</tr>
<tr>
<td>Fixed or Cost Plus</td>
<td>Fixed or Cost Plus</td>
<td>Fixed or Cost Plus</td>
<td>Fixed or Cost Plus</td>
</tr>
</tbody>
</table>

In non-batch projects, any errors or misunderstandings at the start of the project are often overcome at a later stage, with a known and understood impact. In batch projects, the impact is often more difficult to recognize. For instance, a misunderstanding, omission or error during the design stages, which may have appeared to be minor at the time, can have major effects later on in the project e.g. graphics colours or font sizes, naming conventions. A small item overlooked during the design stage can cause a large morale problem and significant cost later on in the project and call into question the integrity of everything and everyone, from the operating system to the hardware platform and from the project manager to the programmer. A fundamental issue with a batch project is the work required at the front-end. The more work put into the design basis of the project, the less effort will be required later on. The S88 approach has helped in this area, by modularizing the process at all levels and by preparing a process model prior to putting the project together.

Why should this make a difference?

Several years ago, an analysis of 100 successful software projects (batch projects are usually software intensive), by Boehm, showed that almost 1/2 of the total project effort (45%) went into the analysis and design phase, as shown below,

![Figure 1 Relative Effort on a Successful Software Project](image-url)
Further analysis of these projects showed some interesting aspects relating to error detection and error propagation.

Almost 2/3 of all errors occurred during the design and analysis phase and more than half of those were not found by the developers themselves. However, the cost of not detecting these errors rises astronomically as the project progresses, as shown in the following graph,

![Figure 2 The Cost of Delayed Error Detection](image)

It is therefore of the utmost importance that the best resources are used in the concepts and design phases of the project and that they are allowed sufficient time to complete and review the task. This may sound obvious and well understood, but key resources are frequently assigned to projects, only to be reassigned before completion of their task due to shifting priorities. Another key team member is an end user representative - ideally a person who joins the team, stays with it and owns the completed project.

We will look at each of the phases of a project, in turn and identify possible pitfalls and how to overcome them.

**USER REQUIREMENTS AND CONCEPTS**

The aim of this phase is to completely understand what the end user wants to get out of the project and for the project team - solution provider and/or end user, to present the initial concepts of the proposed solution, an initial costing and project plan. The modularity and flexibility inherent in S88.01 means that multiple URS's and FS's could be produced for this project phase, for instance based on units and/or phases.
User requirements can be presented in one of several ways. The end user may have developed a User Requirements Specification (URS) himself, he may have employed a consultant or he may ask the solution provider to work with him. It is at this point that the first potential problem can occur.

Say, for instance, that the user has a well developed URS. In the mind of the end user, it represents everything that he needs, so he asks the solution provider to give him a quotation based on the URS.

**Beware!**

There may be items that the end user has completely overlooked, which could cost the project dearly when it comes to detailed design. For instance, an often underestimated item is exception handling. The ratio of exception handling software to “normal code” is usually more than 2:1.

**How do you overcome this?**

It is vital to be able to “play the user’s requirements back to him,” so that there can be no misunderstanding of the intent of any section of his URS. This can be done most effectively by preparing a Concepts Functional Specification (CFS). In this way the solution provider has an opportunity to review the URS, maybe visit the end user location for more information and ultimately provide a document back to the end user for his agreement. This may be a major or minor exercise, depending on how much work has already been done by the end user, but it ultimately shows that the solution provider understands the users needs and the way to achieve them. Another way of addressing this problem is to ensure that the end user stays involved and ideally provides the team leadership or at minimum a team member – to ensure consistency and later ownership.

Items such as exception handling may be addressed by the introduction of standards and procedures during the CFS or the detailed design stage. In this way, all exceptions would be handled by the appropriate standard, removing the uncertainty of a custom approach. Once more, S88 has helped here. By modeling the process and breaking it down into modular components, it is easier for the team to identify areas of common functionality and areas where a more customized approach may be needed. Of primary importance is the need for agreement on operational displays – again, a standard approach can help here.

Having obtained agreement on the solution, the CFS and an initial project plan are signed off and a firm basis for the project is now possible. (The completed CFS also allows a better evaluation of the costs). Many end users, particularly those with a good URS, often feel that this process is unnecessary. However, as indicated above in figure 2, time and again there are items which have been overlooked or misunderstood which can have a major impact.

The signed-off CFS provides the basis for the next phase of the project, the Detailed Design Specification (DDS).

**DETAILED DESIGN**

In the detailed design phase, the CFS is used as the base document. It is expanded to include the details of **HOW** the software will operate. It will include example sections of code, such as prototype phases or screen layouts and identify which release of operating system the applications software will run on. It will also provide a detailed project plan, indicating when key deliverables will be available and when key milestones occur.
The deliverables of the detailed design phase include the Detailed Design Specification, the detailed Project Plan (well written and laid out) and any standards and methods which will be used by the project during the implementation phase.

The detailed design specification explains in detail all of the concepts introduced in the CFS. It contains detailed descriptions of how items will be implemented. In the case of non-standard software, it will include code or at least pseudo-code for each application, operator interface designs, report layouts and abnormal condition management. For standard items, examples of the proposed standards may be included or referenced. At this point the control phases may be simulated and also demonstrated to the end user team – including operators, for agreement.

The detailed project plan lists all of the work breakdown items and their expected duration. It provides an indication of milestones and resources.

The experience of an “expert” on the project, in the detailed design phase, will help with error detection and elimination, but a problem to avoid is the “expert syndrome.” Experts can save a tremendous amount of work on a project by providing design input and reviewing documents and so on. They are wonderful consultants to any project, but often employ tricks, learned with experience, to save time. This may or may not be true. However, if non-standard techniques are used and the expert is called away to another project, now or later on in the implementation phase, the project team will experience difficulties trying to decipher and troubleshoot the solution. A possible answer to the use of these “tricks” is to maximize the use of standard techniques and software modules and to minimize the use of custom code. In fact, many standards are the development of “tricks” which have been tried and tested over a period of time. Once more, S88 has helped here. Batch control engineers have started to speak a common language and the experts who once held sway with their ability to make systems do the impossible have been replaced by S88 knowledgeable process engineers who can help solution providers to map systems to their production processes.

The introduction of “S88 aware” batch control systems has again helped to reduce some of the risk. The software can be designed using a structured technique against a set of project standards to help with coding and testing later in the project. The code is modularized into phases and pre-tested, where possible, so that they can be re-used in different applications. For instance, the code to heat a batch reactor could and should be the same for all reactors. It should be a stand alone module and not embedded in a large piece of custom code, intended to control the complete reactor. In this way, the only change from module to module is the identification of valves to be opened and closed, along with other recipe parameters. The skill these days however, is the “granularity” chosen for the phases and their location in the control system infrastructure. This can have a big impact on the success of a project.

So, what does a standard approach give you?

As shown in figure 1 above, it is essential that maximum effort goes into the detailed design and understanding of the end user requirements. This involves the use of the best resources from both the end user’s and the solution provider’s perspective. The use of standard modules and engineering tools ensures that the implementation phase is very much a “fill in the blanks” exercise. In this way, valuable resources used in the design phase may be used on other projects without jeopardizing the success of this one. As we will see in the implementation and testing phase, it gives less opportunity for unstructured change and for coding error.
At the completion of the design phase, in addition to the detailed design specification, the project plan must also be finalized and signed off. The end user now knows what he is getting, how it will be done, by whom and when. The project scope is now fixed and any changes requested from now on will be evaluated and discussed with the end user, in accordance with the project change control procedure. They will not be implemented until after the testing phase.

**IMPLEMENTATION AND TESTING**

As with the CFS, the DDS becomes the basis for the implementation phase of the project and should ultimately become the basis of the document against which testing is done.

Implementation is the phase where coding, configuration, display building and report building takes place. It is also the phase in which many software projects have come to grief. The reasons are many and varied.

**Let’s look at a few of them.**

The desire with software engineers is always to have the latest and greatest – this is often supported by eager sales managers and sometimes driven by mergers and acquisitions. This may be in terms of platform technology or the base operating system on which the applications will run.

Be extremely cautious about or try to avoid changing any software or hardware platform in mid-project without a thorough evaluation of the potential impact. It may be that the applications will accommodate a change in the operating system: it may not!!

It may also be that the new system hasn’t been fully tested and integrated and you are working with an unreleased version of the core software, on which you are building your applications. Several examples exist of vendors introducing new or "bridge" platforms while convincing the customer that everything is under control - only for the system to lock up or stall at a key point of the implementation.

There will be unplanned changes to the project - try not to accept them and if they are unavoidable work to minimise their impact (using a risk analysis tool) or they will jeopardize the project’s success.

There are several questions to ask yourself here:

- What does the new operating system give to my project?
- Was the new operating system planned for in the design or the risk analysis?
- Can you complete the project and satisfy the customers requirements without it?
- Should you look at alternative systems?

Some of these questions should also be asked before the project even begins.

In another situation, the project team races through the initial stages, accepts the end users URS, does what it considers to be an adequate design and plans to make up for any problems in the implementation phase. After all, “we can modify the code!”
In this situation the “gotchas” come into play. For instance, as mentioned earlier, the possibility of exception handling. It may have been mentioned in the early stages of the project, but not quantified. Now it may double or triple the implementation phase schedule and costs.

Now – this is where even S88 may not help, as many of the S88 aware systems need to have the code laid out in certain ways for the system to work at its optimum.

**What about project personnel?**

If the design team is different from the implementation team and they have not been in communication, then you have a recipe for disaster. The implementation team doesn’t know what the design team intended and may go for the option which makes more sense to them, but doesn’t satisfy the end user requirements. Changes to key personnel such as lead engineers and even the project manager can have major effects on the project. Again – the standards suggested by S88 have helped to reduce the effects of this problem. By using a standard approach to project design and implementation, it has become easier to move people in and out of a project. However, as with previous projects, if this is done too frequently, problems will still occur.

Finally, beware of the “add-in.” In a normal project, it may be relatively easy to add an extra valve or piece of pipe. In a batch project, a small addition, such as an extra control loop can cause major problems, unless it is analyzed and its effect quantified. For instance, all of the displays in an area may need to be rebuilt or the exception handling criteria for the affected area may need to be re-evaluated.

Most of these can be dealt with effectively with the use of standards, tools and a well thought out design. In addition, an effective project “control system” will ensure that the project stays on track. This will be dealt with later.

The use of engineering tools in the design phase, coupled with the use of standard, pre-tested software modules in the implementation phase, should significantly increase project controllability and reduce the margin for error.

The testing phase of a software project has traditionally been the time for a “final clean up of the code.”

Again, with all of the work done up-front in the design and with the use of standard software modules, the testing can concentrate on small amounts of custom code, which will inevitably have been built and are the biggest potential source of error.

**CLOSE OUT**

When the project has been tested against the user requirements and signed off by an end user representative, the project may be closed out. The software is then handed over to the end user. It is important to establish ownership of the software. Does it belong to the end user or to the solution provider? Again, if the software modules are from the solution providers “library,” it is much easier to identify the owner. Any new modules developed by this project and of potential future use to the
solution provider may be the subject of a further negotiation – intellectual property rights have become a major “bone of contention” in recent years.

So now it’s all over. Or is it?

Beware the warranty requirements on the software supplied. Study the contract and make sure that you have the capability to get support for the delivered software over the life of the warranty and there is on-going “bug fixing” capability. This is also made easier by the use of standard software modules, in that more than one person may be able to provide the required back up.

Hence, with a structured project procedure and the use of standard design and implementation techniques, it is possible to keep a software project under control.

An additional aspect will occur in training. All software projects require training in the area of operation, applications development and system support. Some of this will inevitably be included in the original estimate, but there may be issues when the total training requirement is evaluated.

Finally, the control exerted on the project will determine how successful it becomes. There are several techniques to bear in mind to help in this aspect.

HOW TO REDUCE THE IMPACT OF CHANGE WITH PROJECT CONTROL

Risk and Opportunity Assessment

In any project, but especially in software projects, there should be an assessment of the potential risks involved and any corresponding opportunities. The impact of the risks, the probability of them happening and the proposed mitigating action to be taken should be assessed at the beginning of the project and continuously re-evaluated as it progresses. In this way there should be no surprises.

If there is a risk that a new version of the operating software is going to be forced onto the project at some stage, this can be taken into account in the design.

Likewise, the potential opportunities should be analyzed. Training was an opportunity indicated in the Close Out section. Another opportunity may be additional applications support once the system is on site, to optimize its performance.

Contract Management

Establish what each party is responsible for, both technically and commercially. Make sure that the initial contract is acceptable and that each party is able to meet the contractual terms. For instance, beware of the phrase “will use the latest revision of system software.” The introduction of a new software revision could produce major problems later on in the project, unless of course the risk has been analyzed and assessed.
Quality Plan

A quality plan should be produced at the beginning of the project. It indicates such things as how project status reporting will occur, how revision control will be effected, what change control methodology will be used and so on. This will normally be a standard document.

Reporting

Monthly project status reports should be produced, indicating accomplishments during the month, plans for next month and any issues. It should also identify the current status of the risks and opportunities and what action is underway to mitigate them. The status should be re-estimated each month to continuously calculate what has to be done, not what has been done. This prevents the “95% rule”, i.e. a project makes “good progress” to 95%, then each following month sees no progress from then on, so the project remains effectively at 95% complete.

Revision Control

Ensure that all documentation and software is under tight revision control, so that the whole team is working with the correct revisions at all times. It is not unknown for testing to be done on the wrong revision of software or to the wrong revision of the URS/CFS.

On every copy of the software and documentation, the revision number and date should appear on all pages. On the cover page, an indication of the changes from revision to revision should be documented, including the author of the revision.

Change Control

Changes will always happen to projects and batch control projects are no exception. The problem comes when the changes are incorporated into the project without a change control system in place. The initial aspect of change control is the risk assessment process. This will identify the risk of some changes being made before they happen and a plan will be in place. It is important to note that these changes still need to be evaluated against a formal change control procedure.

These should be evaluated for their impact to the project cost and schedule. A change notification is then formally communicated to the end user representative for his approval. The change should only be incorporated after a signed agreement between the end user and the project manager. As mentioned above, NO changes, either planned or unplanned, should be accepted after the project scope is fixed.

And…beware “creeping change”….as it WILL bite you.

Finally, some common sense rules to apply to software project management:

GOLDEN RULES AND PROVERBS

- Always use a structured (S88) approach and break the code into manageable, reproducible pieces.
The more time spent at the front end of a software project, the easier the implementation.
Always carry out a risk assessment prior to and throughout a project.
Software is easy to change, but only in the sense of a “Rubic Cube.”
Do not make uncontrolled changes to a project.
Project progress is inversely proportional to the number of uncontrolled changes in the project.
In the event of a problem in implementation, look for the item considered the least important in the earlier stages of the project.
If it isn’t recorded - it doesn’t exist.
If it is recorded - it may not be correct.
If everything appears to be going well - it probably isn’t.
If it isn’t broken - don’t fix it.
Uncontrolled projects stay 95% complete forever.
S88 has helped to control some of the issues with batch control projects – but it is not a panacea and cannot replace good planning and project management.