IT integration problems and the nature of an Integration discipline.

Summary

IT projects experience major problems because of delays integrating the components into a working solution. This is a recurring theme, one that leads to vast schedule and cost overruns and complete project failures. These problems are an inevitable consequence of the way projects are approached. In particular they are inevitable consequence of the lack of disciplined approach to achieving early integration, the lack of a discipline of Integration itself. Changes to the way programmes tackle these issues are needed to address this. There is a need to establish and apply a discipline of Integration if continual failure is to be avoided.

Table of Contents

1. Introduction

This article is the first in a series of articles on integration failure and on the discipline of Integration. It provides both an introduction to the topic and an executive overview. The overview (a) describes the detrimental impacts that poor integration delivery has on IT programmes and (b) introduces the discipline of Integration as a way of addressing these problems.

The content is based on wide experience in the management, delivery and assurance of complex large scale IT programmes across a variety of sectors within the UK . It reflects many years of observing what goes wrong and what works well and of having to fix things that failed. It describes the recurring problems and failures; putting a spotlight on their endemic nature. It outlines Integration; a consolidation of the body of knowledge of what works well into a coherent discipline that tackles this issue.

2. Recurring nightmares; poor integration

The starting point of this article is to emphasise the recurring pattern we have recognised. It is certain that we are not the only ones that have recognised it. Many readers may think that there is nothing new in these sections. However what we are doing in is emphasising how common and how endemic the problem is and how much waste and how much failure it causes. Perhaps the unique thing that this article does is shout about it so loudly.

A story, well a sorry story, the lack of integration

Consider the following questions. How many times have projects been delayed because when the various elements of the system are brought together it takes a long time, too long, to get them to interact to do anything useful? We don't mean working perfectly; we mean getting the mainstream functionality working so that it can then be tested. How many times have you, as a programme manager or as a member of the development or test team, experienced frustrating delays wasting precious schedule time and resource waiting for the basics to work?

Look into what is happening and we find that there are multiple fundamental breakages in the operation of the system. Interactions between components fail. Failure often occurs at the boundaries between systems, technologies or organisational units. These breakages are time consuming to resolve. Do you recall waiting whilst someone, and often it is not clear who this someone should be, figures out what is wrong, waiting whilst there is a long debate over who is right and who is wrong and who must change and waiting whilst multiple attempts are then made to make the right change?

The story progresses. At last there is success; a change clears this one basic problem that has been blocking us for so long. The fix works when the operation is tried; only for us to run into the next problem that no one foresaw and no one has yet started to worry about. The cycles and delays repeat. Sequentially, one at a time, these problems are found and cleared. Eventually the last one is cleared; now we can actually start to progress; but it is too late we have missed a vital date, missed a release window, stakeholders have lost confidence; there is a crisis.

Have you had a similar experience?

Now consider this. Do you recognise this pattern? Have you experienced it yourself? We have seen it on too many occasions. Unfortunately it is a recurring feature of both complex and not so complex projects. Our own experience includes examples such as:

- A system that took four months from the initial delivery of the components to the first provisioning order passing up and down the processing chain in a recognisable form.
- Web-site single sign-on projects that missed their dates and fell out of the system releases they were scheduled to deliver in. They simply did not and could not work; the components did not fit together. The solutions had to be re-designed and reworked as each problem was uncovered.
- Major releases delayed for weeks incurring very significant additional cost to retain large project teams. The cause? A key part of the release failed to integrate. The changes could not be made safe to allow the main release to go ahead[1] and so the entire release was delayed.
- An attempt to implement online versions of call centre processed sales orders where it was found that there were fundamental incompatibilities in the way the new channel and the call centre system modelled the products. The results; a complete, expensive and risky emergency re-design of the way the two systems were to work together.

[1] Depending on the success of a high risk change in order to be able to deploy a larger release is itself a fundamental mistake and a lack of risk management. However this is a wider topic not covered in this note.

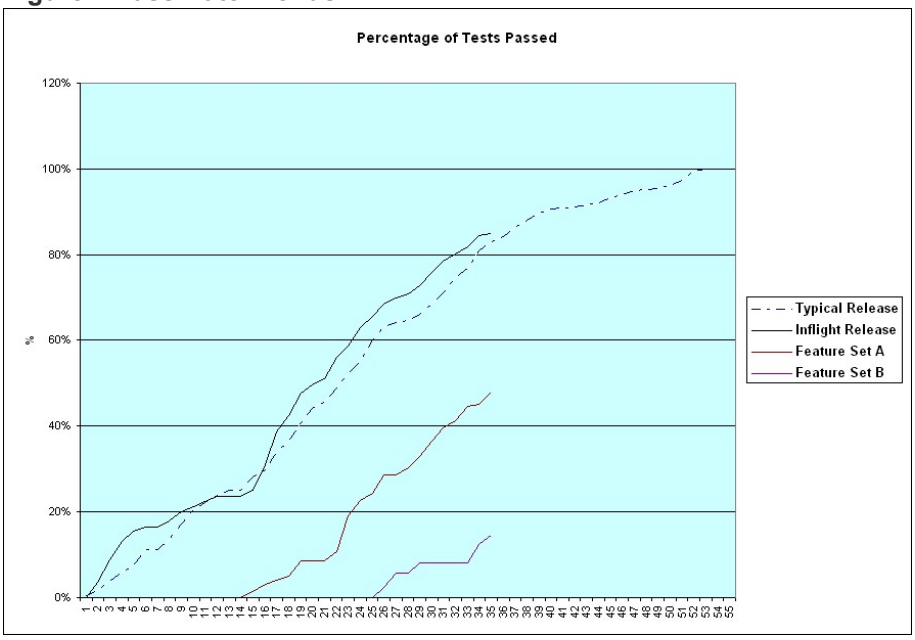
Our experiences are not unique. Issues of this nature and the damage they cause are widespread. Their end effect can be catastrophic. This phenomenon is widespread within the IT world and we are certain most readers will have encountered it.

Symptoms of a project with poor integration.

The symptoms of this disease are delays; ones that often manifest themselves as low test pass rates. Figure 1 shows some data from a representative project. The upper two plots show the trend over time for a typical release of this system and for the release that is in-flight. The aggregate performance of the in-flight release is following the typical path. However the two lower plots show the individual status of two key elements contained within the release.

Both of the highlighted elements suffer from integration problems. Feature set A lost at least fourteen days before it started to progress. More severe are the problems of feature set B. This has lost twenty five days and even after it started to progress its rate of ascent is below the norm.

Figure 1 Pass Rate Trends



Consequences of poor integration.

The consequences of integration problems depend somewhat on the way the organisation reacts when they are encountered. Generally they are never good. A common consequence is delay to the releases date. Alternatively the release date may be held but the problematic payload made safe and dropped or partially dropped. Less explicit but common is a reduction in the quality of the work done and the product delivered due to a compression of the windows for testing and fixing it.

Immediate consequences of poor integration are seen within the programme itself. Ultimate consequences manifest themselves in the delivery that the programme makes and in subsequent operational issues and improvement requirements.

Immediate consequences of poor integration include:

- Compressed timescales for down stream activities. They start late because there is nothing to work with at the time they should start. Their end dates are not moved in an attempt to prevent slippage. An effort is made to do work in a shorter window. Smaller windows often means removing task, less attention to detail and less

time to consider and recognise what is really happening. Overall there is a degradation in the quality of work performed.

- Compressed timescales will lead to extended working hours. The phrase “working 24 by 7" will start to circulate. Extra resources may be poured in. This may incur direct financial costs for the programme in terms of overtime and contract resources or it may indirectly cost the wider organisation as resources are diverted from elsewhere.
- Additional resources will be added to the project. Resources that do not have the depth of knowledge of the existing team members. The knowledge of the team is diluted with a detrimental impact on quality.
- Disruption, uncertainty and rework. Poor integration not only leads to late starts it is inherently disruptive. It leads to late starts on partially integrated unstable solutions that are still evolving. This has a disruptive effect causing tasks to fail, causing uncertainty and invalidating whole areas of completed work due to subsequent changes.
- Either end date slippage or a major reduction in delivery scope. Despite teams being asked to deliver in compressed timescales and doing their best to do so tasks will slip and new tasks will arise due to instability and change. It will not be possible to deliver the full scope on the target date to a production quality. Compromise will occur and one or more will suffer.
- Delivery quality will suffer. No matter what adjustments are made to scope and timescales the quality of the delivery suffers. The disease spreads and affects downstream activities. Some compromises in quality will be consciously factored in as part of the recovery plan. Beyond this there will be a further deterioration arising from the disruption caused by the disturbance of the planned cycles.

Those were the immediate consequences suffered by the programme attempting the delivery. Beyond that are the ultimate consequences experienced by end users and programme sponsors. These take the form of:

- Late delivery of possibly partial solutions reducing the business benefit. Potentially worse; a total project failure causing a complete loss of a business opportunity.
- Increased direct programme costs arising from the use of additional resources in recovery attempts and from extensions in the duration incurring cost for the overall project team.
- Additional operational effort and maintenance work to deal with quality issues in the delivered solution.
- Poor customer experience arising from reduced functionality and quality problems within the delivered functionality.
- Follow-on project costs incurred completing the delivery of the original scope of the programme at a later date.

In conclusion the consequences of integration problems are bad; very bad. The consequential costs far outweigh the costs of addressing the problem; it is just that the problem is not recognised and even if it is people do not really know how to tackle it.

3. Integration the missing discipline

Having highlighted the issue that is affecting IT programmes the other key aim for this article is to detail our assessment of the cause of the issue and our view on the road to an improved situation. This section does that. It gives our analysis of the root cause; a position drawn from our experience. It makes the case for a new discipline to tackle this issue and it provides a high level manifesto for this new discipline.

The root cause of the problem.

Where do these problems come from and why do the same sorts of problems hit project after project? Underlying it all is their lack of integration. Whether this is a green field project delivering a new complex solution or a change that is restructuring or extending an existing solution there is a need to get diverse elements and systems to work together in new ways. This is a need that is rarely addressed in an effective way.

These constituent parts will generally have been developed by different teams, perhaps using different technologies / packages and almost certainly based on different understandings and assumptions. Often there is little contact between teams. Unsurprisingly when we simply plug them together the resultant system tends not to well if it works at all.

Extensive remedial work is normally required. Yet we find that organisations are not effective at handling multiple widespread fundamental breakages in new features; the fix processes are uncertain, error prone and take a long time. Often it is not clear whose job it is to solve these problems. This challenge frequently crosses internal organisation boundaries if not inter organisational commercial boundaries. There is no planned approach to integration. No one is prepared. The process that follows is often drawn out and fraught.

We believe the root cause is a lack of an appropriate focussed effort to ensure things become integrated quickly. This represents a lack of a fundamental discipline in the IT programme domain.

The case for Integration.

Projects will continue to suffer delays and waste if they do not take measures to address these issues. There is a need for an effective approach to tackling these causes of delay. This approach is something we will refer to as Integration. Here Integration has a capital “I”. Capitalisation denotes that Integration is an activity, or set of activities, designed for this purpose rather than a general objective.

Here the word has a more specific meaning than is normally associated with it. We use the term in the same way as terms such as Programme Management, Design, Development and Testing are generally used. It is used to denote a recognised set of responsibilities and activities that make a planned contribution to the outcome. We use Integration to denote the set of activities that deal with prevention and rapid cure of the sorts of integration issues described at the start of this section.

The big problem is that many development processes and programmes do not recognise the impact of integration issues and do not build in Integration. They do build in disciplines such as Design and Testing but Integration is not a recognised part of the process. Projects go straight into testing hoping integration defects can be fixed as a result of finding problems during test. For many reasons this is inefficient and ineffective. These projects suffer delay, cost increases and produce low quality outcomes.

Integration in a nutshell.

Our model of an Integration discipline is summarised below:

Integration is the use of virtual teams, drawn from the various component supplier teams, to execute a planned set of tasks and reactive activities to prevent and, where prevention fails, eliminate as quickly as possible integration issues. Virtual teams bring together the knowledge necessary to prevent and solve problems. Virtual

teams by their nature reduce friction between separate component teams. The objective is to achieve effective integration as early as possible. The approach is to focus as much on prevention as on cure. The ethos is to expect problems and to be ready to deal with them.

That is Integration in a nutshell.

The Integration Manifesto

This article does not aim to spell out in detail the aims or approaches of this discipline; for sources of more detailed information see the section at the end of this article. However this article does aim to give the reader a clear understanding of the shape and ethos of the Integration discipline. The "in a nutshell" description needs a little expansion in order to achieve that. We attempt to do this in the from of a manifesto for Integration. Hopefully this next level of definition will suffice to give the reader enough of an insight to decide whether it is worth undertaking further reading. The manifesto of Integration consists of twelve fundamental principles:

The Integration Manifesto: Edition 01	
01	Integration has a primary objective of accelerating the transition to an integrated state. The aim is the earliest possible achievement of a state that permits down stream activities to start and progress without disruption. In this way the momentum of the programme is sustained.
02	Integration is an explicitly planned part of the programme of work. As such it has allocated objectives, is staffed, has leaders and whatever resources are required. There is plan for an explicitly defined intended scope of work. It is not an ephemeral best endeavours activity done in peoples spare time.
03	Integration applies the rule that prevention is better than cure. The ethos is not to wait and see what gets delivered; instead it is to apply influence to improve the starting state when a delivery is made.
04	Integration is a technical activity. It is not a black box activity; it focusses on the technology, on what it has do and on how it operates.
05	Integration does not attempt a tasks for the first time when it is actually on a critical path. Unnecessary delays are not to be tolerated. This particularly applies to delays on Integration tasks that must be completed before down stream work can start. Therefore preparation and rehearsal are vital. In advance of a task being due the approach, techniques, methods and tools for doing the task are prepared and the task and all sub-tasks are rehearsed.
06	Integration fixes problems it does not find them and then leave them to others. When a problem is found the Integration team identify the solution and drive its delivery. The team is empowered to do this. Integration is not simply a testing function.
07	Integration expects many changes and many problems. Things always change and things always go wrong. Integration prepares ways that will allow it to continue to progress in a fluid situation and thus minimises the impact on the end date.
08	Integration focusses on the greatest integration risks. It actively seeks out these risks in the early periods of a release and then works to prevent the risk from maturing into an issue for the programme.
09	Integration starts early. Prevention requires up front engagement and intervention. Preparation and rehearsal requires time. An integration team cannot be stood up two weeks before a delivery is to be made for integration.
10	Integration uses collaborative working by all contributors. Delivery uses virtual teams drawn from all suppliers of components in the items to be integrated. It is collaborative. It is a joint responsibility of all suppliers to make the integrated item work. This nature is embodied in the use of virtual teams. They integrate the organisation before trying to integrate the solution.

The Integration Manifesto: Edition 01	
11	Integration is not dogmatic about who performs it. Teams are built from the most appropriate parts of the organisations involved in the programme. Similarly leadership is taken from the most appropriate source.
12	Integration uses experienced people. Experience and competence are critical requirements for team members. Members need to contribute the body of knowledge and capabilities of their own organisation. Members need the respect of their peers. Effective integration can not be performed by a team of junior personnel.

Experience shows that compliance with these twelve principles is the effective route to mitigate Integration risk and to avoid programmes being derailed as a result of integration problems. Where we have seen these principles applied we see a smoother quicker progression to a stable state able to support down stream activities. Where they are ignored we see projects in trouble.

Establishing the Integration discipline

One final topic for this section is the matter of introducing Integration and establishing it within a development process. The difficulty and complexity involved in doing this should not be underestimated. The collaborative nature is relatively unique. Integrating the organisation may prove as difficult as integrating the system. This challenge increases even more when the overall organisation spans commercial boundaries.

Integration will not become an effective part of the process by magic. It will require active support from sponsors who both understand it and believe in it. A roll out plan is required and personnel will need briefing and ongoing support. People do not become fully engaged in new ways of working overnight.

Care needs to be taken to incubate the practices in appropriate environments before exposing them to the more hostile ones. It may be tempting to apply Integration at full scale to prevent the next major strategic release failing in the same manner as the previous one; but it is not a good way to learn to do something for the first time. Far better to develop experience on smaller programmes and incrementally progress into the wider arena.

Most important of all is the need to recognise that successful adoption will have to be incremental and flexible. At each stage the team ensure that the fundamental principles are being applied whilst evolving the detailed practices to align with the culture and dynamics of the organisation and the challenge of the next programme to be tacked. A local variant of Integration will emerge; one that matches the need of its environment.

4. Consideration of counter arguments

Having made a case that there is a problem and proposing a solution it is worth considering some of the counter arguments that might be made to suggest that there is no real need for something new. This section considers some potential counter arguments and responds them.

But aren't we already doing Integration?

Firstly we will consider the argument that Integration is already a well established and widely used practice. The term Integration or variants of it is in widespread use within the IT development world. Phrases like "Integration Testing", "Continuous Integration" and "Component Integration" abound. Organisations contain "Integration Teams" and plans or at least development models contain an "Integration Phase" or an "Integration Test Phase". With so much "Integration" around how can we say that we do not tackle Integration?

The answer is that effective Integration, executed in a way that avoids the waste and delays described earlier, goes well beyond the more common activities mentioned above. Whilst it can encompass one or more of these activities it fundamentally differs from the typical "integration" activity in having a far greater breadth. It has a different focus and approach. The activities described above are but one part of the Integration activity.

A common characteristic of the "integration" activities presently practised is their place within the Build – Deploy – Test pattern. Generally integration is a part of the Test activity; the emphasis is on assessing how integrated the system is and on detecting breakages. It is a "detect what is not integrated and measure overall status" mind set rather than a "lets become fully integrated as soon as possible" activity. Whilst "detection" if done well (not always the case) can mean earlier, though not necessarily early enough, fixes this in itself does not significantly accelerate the journey towards becoming integrated.

Integration, with a capital "I" aims to get us from the state where the system is not integrated to the state where it is integrated faster. It is not simply about identifying what is broken and then letting fixing take its course; it is about getting there as soon as possible.

Well that was a waste of time wasn't it!

Integration, like testing, can suffer from the argument that if it didn't (appear to) contribute anything then it wasn't necessary and should not be done next time. More to the point if it is a success there will be very little to show for it; apart from a smooth progression into the subsequent phases and its contribution can be taken for granted. You might start to hear the argument that spending time and money on Integration is wasteful and that if we hadn't done it not only would things have been the same but we would have started the next task earlier. How short some memories are.

Given these characteristics and the surfacing of this argument; how is all that time, effort and money to be justified? How is the need for Integration to be argued. A mature intelligent management environment that understands risk is fundamental to reasoning about this in a sensible way; so on that basis for a number of you the battle may be already be lost. Once established and operating well Integration is a risk mitigation activity. The costs involved need to be offset against the impact if things do go wrong and these are not concrete. Hence the need for a mature environment that understands risk. There is no other way of winning this one; apart from losing it once or twice and letting the consequences speak for themselves.

5. Concluding this article.

This article concludes with a recap of the key points and with a guide to locating other articles and information on Integration.

A Recap

The key points of this article are:

- There is a recurring pattern of solutions failing and of major cost escalations within IT programmes. One source of this is that when sets of components forming a system are delivered they do not integrate into a stable working solution in time to permit down stream activities to deliver effectively.
- The principal reason for this is that the programme assumed everything would work together reasonably well within a reasonable timescale without any specific intervention to make certain this happened. This is a recurring erroneous assumption.
- As a result there are no plans to prevent or to cure significant integration issues and so they occur and then have serious detrimental impacts.
- Overcoming this problem requires the insertion of an extra discipline into the programme. This is the discipline of Integration.
- A key characteristic of Integration is the use of virtual teams drawn from the disparate suppliers of components to be integrated. This integrates the organisation ahead of the technology and so aids problem prevention and removal.
- There are twelve items identified in the Integration manifesto. The more completely the principals embodied in each of these are applied the more effective Integration will be.
- Adoption of Integration should be incremental. Big bang approaches are likely to fail. Within each organisation Integration should be adapted to align with the nature of the organisation itself.
- Generally Integration, as it is defined here, is not in use. Activities that are described in terms using the work “integration” at best touch on part of the Integration space and tend to apply few of the fundamental principles of effective Integration.

Further Information

Readers with an interest in looking at the concept of Integration in more discipline may be interested in the following articles that explore the topic in more detail.

Integration; adopting a new discipline.	The second article in the series explores the nature of an effective Integration approach and the four main challenges faced introducing and sustaining an effective Integration effort.
Integration; practices and processes.	The third article starts to focus more on what needs to be done when and how to go about doing it. At a high level this article starts to provide a handbook for integration supported by an explanation of why things should be done that way.

Background Information

This section provides background information on SQC and on the author of this article. Readers requiring more information can visit the company web-site at www.sqc.co.uk or send a request by email to enquiry@sqc.co.uk.

Author Biography

The author of this paper is the principal consultant at SQC. His career has focussed on delivery and assurance of complex IT solutions and high risk systems. His professional career began in 1985 and over the years has spanned development, testing, test management, test automation, delivery management and programme assurance. He began to focus on testing in 1991 and spent over a decade as a lead practitioner in this field. In the four year period to the end of 2009 he was the head of integration and test for the retail arm of the principal UK telecommunications provider. In this role he not only built and managed a function with a multimillion pound annual test budget but also served as one of the main programme leads shaping and managing complex IT programmes worth hundreds of millions of pounds.

His career has encompassed a great diversity of system types, development practices, project characteristics and organisation types. It has seen him leading both technical definition and effective delivery of a diverse portfolio of activities spanning system definition, design, implementation and test. The latest phase of his career has seen him shaping very large scale IT delivery programmes, providing independent programme reviews, driving recovery programmes and building and leading an enterprise wide integration and test function.

SQC

SQC originated as a supplier of software testing services; a provider of consultancy, test management, test delivery, load testing and test automation. SQC started serving this market in 1991. The organisation's expertise and field of operation has broadened over time to include the wider programme delivery domain. Today SQC's expertise spans from Programme Assurance through Test Management and Test Delivery via specialist test automation to technical testing. In these fields SQC can provide leadership, delivery management, service delivery, associated technical services, oversight, consultancy and training.