Integration in practice; what needs to be done and how to go about doing it.

Summary

There are major problems being experienced in large scale IT programmes due to poor integration and hence there is a need for a discipline of Integration that will address these problems. This discipline requires a framework to organise it efforts. A five stage framework that is suitable for managing Integration contains the following stages:

- Preliminary Assessment
- Team Mobilisation and Team Planning
- Pre-delivery Preparation and Prevention
- Target Integration
- Maintenance and Support.

Initial risk identification and allocation of work to separate teams is done by the core team during Preliminary Assessment. The separate virtual teams then perform the detailed risk assessments and planning of work to deal with the risks. At the end of Team Mobilisation and Team Planning the plan for the Integration effort exists.

There is a heavy focus on prevention of problems and on preparation for the eventual hands on integration work that will have to be done. This takes place in the Pre-delivery phase. Prevention uses both avoidance and elimination. Avoidance aims to prevent people introducing errors into the concrete realisation of the system; into the source code and schemas that govern its run-time operation. Elimination aims to enhance the business as usual development review and test activities to detect and fix those that do appear before components are delivered for Integration. In parallel with the prevention activities preparation for the actual integration occurs. Tools and techniques are prepared and tested. Processes are rehearsed. The aim is that nothing is done for the first time when the integration is being done for real.

Target Integration takes the delivered components, evaluates the integration and deals with any issues. There is a very heavy emphasis on rapid coverage to learn as much as possible as quickly as possible and moving to a working state as soon as possible. This is reflected in objectives such as day-one feedback and rapid issue resolution.

Once an item is integrated the owning Integration Team are responsible for support and maintenance. They aim to ensure that neither integration issues, nor misunderstandings or lack of knowledge of the integrated item itself delay or disrupt downstream activities. This avoids a successful technical integration being derailed because the teams doing subsequent activities do not have technical knowledge. The Integration Team is as responsible for the item being used effectively as much as it is responsible for the item working.

Table of Contents

1. **Introduction** . . . 3
2. **Recap; why Integration?** . . . 3
3. **Recap; so what is Integration?** . . . 3
4. **Integration; here is an "in a nut shell" guide.** . . . 6
   - The Participants . . . 6
   - The Stages . . . 7
5. **Stage-A: Preliminary Assessment** . . . 7
   - Approach . . . 7
   - Objectives . . . 7
   - Timing . . . 8
   - An aside; full Convergence depends on Preliminary Analysis . . . 8
   - Principal Tasks with Preliminary Analysis . . . 9
   - The End Game . . . 9
6. **Stage-B: Team Mobilisation and Team Planning** . . . 9
   - Approach . . . 9
   - Objectives . . . 10
   - When is does this have to be done by? . . . 10
   - Mobilisation; forming and tasking the teams. . . . 10
   - Team Planning Activities . . . 11
   - The End Game . . . 13
7. **Stage-C: Pre-delivery; preparation and Prevention** . . . 13
   - The importance of the pre-delivery stage. . . . 13
   - Approach . . . 13
   - Objectives . . . 14
   - Generic Team Preparation . . . 14
   - Preparation for Target-Integration; diagnosis and cure . . . 15
   - Prevention . . . 17
   - Prevention: Avoidance Measures . . . 17
   - Prevention: Elimination Measures . . . 22
   - The End Game . . . 23
8. **Stage-D: Target Integration** . . . 23
   - The Approach . . . 24
   - Objectives . . . 24
   - Evaluation and Issue Detection . . . 25
   - Diagnosis & Characterisation . . . 29
   - Workaround and Correction . . . 31
   - The End Game . . . 33
9. **Stage-E: Integration Maintenance and Support** . . . 33
   - Approach . . . 33
   - Objectives . . . 33
   - Integration Maintenance . . . 33
   - Support . . . 34
   - End Game . . . 35

1. Introduction

This is the third in a series of five articles on the topic of Integration. The first article [reference 1] discussed how IT projects experience major problems as a result of integration problems. It went on to introduce the concept of having Integration as an organised, planned and staffed practice focussing on prevention and rapid cure of integration issues. The second article [reference 2] addressed the nature of the Integration discipline and the four main challenges that would be faced introducing and sustaining the use of Integration. This article now goes on to discuss the tasks and activity that make up an Integration effort and the framework that glues them together into a cohesive whole. It also outlines some of the techniques that can be used during Integration.

Readers who have not done so are advised to read these two articles before proceeding with this one.

This is quite a long article. It starts with recaps over a couple of topis. Firstly over the reason why we need a discipline of Integration and secondly over the nature of the discipline. There is then a short “in a nutshell” guide that introduces a five stage model to shape the Integration effort. Each of the stages is then described in more detail.

The descriptions of the two main operation stages, Pre-delivery work and Target Integration, include within them descriptions of the types of methods and techniques that can be used by the teams carrying out Integration. They thus not only define the organisational shape and objectives of these stages but also illustrate the sorts of things that can be done to prevent, eliminate and deal with integration faults.

2. Recap; why Integration?

The aim of integration; it is “to become fully integrated as soon as possible” and this is so that downstream activities that can only progress effectively once the solution is operating can proceed. Failure to integrate creates a bottleneck with problems blocking everyone. We want to do Integration because without it timely progress generally does not happen; things take too long and they derail projects.

Why do we need to address this issue with a new discipline of Integration rather than incremental changes on existing practices. Integration is required because the way medium and large scale IT delivery is generally organised combined with lack of focus and investment on mitigation of integration risk is the major origin of the problem that needs solving.

We do need to be beware of blurring boundaries though; Integration is not about getting the system to pass every test; it is about getting the system into a state where the majority of operations work in a recognisable form and the majority of tests can be run and give some kind of meaningful information. We want to do this quickly because failing to do so means missed deadlines and massive cost increases.

3. Recap; so what is Integration?

The job of Integration is to get the system working well enough to allow downstream activities to progress effectively and efficiently and then to support these downstream activities where they do encounter or discover integration issues. It does not encompass the downstream activities themselves, it does no focus on issues contained within individual components (other than where they are the cause of an integration issue) and in its current guise it does not address detailed testing of the integrity of the integration. It facilitates keeping the programme moving at pace and avoids the sudden loss of momentum as bits of the system come together, do not work and leave teams standing around with little to do as days and weeks are lost.

The discipline of Integration embeds into work plans a set of preventative and curative tasks along with the preparation tasks that will enable these primary tasks to be executed. These tasks all focus on the integration

risks. They intercept development activities to prevent integration faults and where this fails they detect, diagnose and eliminate issues as quickly as possible. Successful Integration depends upon collaborative working across supplier teams. However it is a planned and managed operation and is not done on a best endeavours basis. It depends upon creating a virtual team organisation that draws all the necessary skills and knowledge together. Success depends upon integrating the organisations before attempting to integrate the products they are supplying.

The essential features of an Integration discipline are embodied in the principles of Integration; these were introduced in the second article of this series [reference 2]. The more effectively these principles are applied the more effective will Integration be. The subset of principles. that directly influence and constrain the processes and practices of Integration are repeated in Table 1. The exclusions are generally ones that address the political and cultural conditions, organisational approaches, mobilisation and adoption.

Table 1: Principles that influence processes and practices; what is done and how to do it.	
P4	The remit of Integration is to get things working as rapidly as possible; not to wait till a system with problems is delivered, find a problem and then to wait for someone else to diagnose it and address it.
P5	To be effective Integration has to focus as much on prevention as it does on detection and cure. Without effective prevention the frontline integration activities will become swamped. That said; expect to do and prepare for a lot of curing.
P6	So; whilst prevention is a priority there will always be problems and the so Integration must plan and prepares to make effective progress when problems are encountered. The Integration team expect to 'debug' and prepare for this. They expect to need to workaround problems until they are fixed. They don't expect things to go to plan but they need a plan; they need a plan that is inherently flexible and adaptive.
P7	Integration is a technical activity that requires a diverse team with comprehensive understanding of the technical architecture and of the technologies in use. The team needs to be self contained and able to deal with technical issues rapidly. It cannot wait for external support on key technologies or design issues as this would cause delay.
P8	The team needs the right skills, tools and equipment to operate effectively. Missing skills and inappropriate dependency on ad-hoc arrangements or on external support will introduce crippling delays. Failure to provide or prepare tools in advance will result in further delays when they are required.
P10	Preparation is everything. The team must be a team and not a collection of individuals by the time the real works starts. Knowledge must be developed and shared. The operation must be planned in advance, all necessary infrastructure and tools must be in place and wherever possible the operation should be rehearsed.
P-B5	Ensure teams are teams in more than name – gaining the massive increase in effectiveness that a team has over a collection of individuals.
P-B6	Mobilise and sustain for the full campaign – to get effective delivery of preparation activities and follow-up activities which are essential if the core integration is to be a success.
P-B7	Align small teams with the risk profile – to allow the team to focus and to provide each with a clear set of objectives.
P-D1	Integration as a whole needs a clear remit; one that is clear to all involved and to the wider community participating in the programme. It is important that it is very clear what it covers and what it does not cover. The remit needs to be tight and focussed on preventing Integration issues sinking the programme.. Integration should not become a name used for a wider ad-hoc collection of activities. There should be a written remit or brief that summarise what Integration is and what ir is not.

Table 1: Principles that influence processes and practices; what is done and how to do it.	
P-D2	Integration task selection must focus on risk. There must be explicit identification of he sources of integration risk. They must be ranked. Tasks to tackle them are generally planned starting with the highest ranked risk.
P-D3	Task selection needs to address prevention and cure. As much if not more attention is given to actions that will prevent problems being present in the delivery as on actions that will detect and remove problems.
P-D4	Identify preparation tasks in detail. Tasks can not be executed effectively and quickly without adequate preparation. Preparation needs to be done in advance; not when the primary task is due. Identifying what to do must also identify all of the preparation required to support the task.
P-E1	Clearly scope the work to be done. Do this early enough so that on-time delivery can be planned and sufficient resource applied.
P-E2	Partition the work into separate final deliverable items. Where appropriate identify intermediate deliverables that contribute to the final deliverables. Allocate a value to each item so aggregate output can be tracked.
P-E3	Define objective measures of the completeness and quality of each deliverable.
P-E4	Establish, maintain and clearly publish the time window within which the work should be performed. This is a joint responsibility between the people doing the work and the sponsor of the work. Have off set windows for different aspects of the work. This provides clarity on start dates; people know if they have not started that they are behind. This provides clarity on end dates; people can tell if they are going to be late.
P-E5	Schedule tasks to achieve the work to obtain a plan. Plot the forecast trajectory of the cumulative value provided by completed deliverables. If the forecast does not show 80% of the value being achieved by 50% of the way through the window then either reschedule tasks or accept that the target duration is unlikely to be met. In the later case there is a need to revise plans at the parent level.
P-E6	Track actual completion value against the forecast and intervene on under performance.
P-E7	Have a competent and empowered individual responsible for delivery management. It is there job to confirm the schedule is one likely to be met, to track divergence from the plan, to ensure those doing the work do not fail to notice divergence and to intervene should intervention be required to avoid slippage in schedule, coverage or quality.
P-F1	Integration requires effective delivery management of the task set making up the Integration effort. This must cover preparation, prevention and execution. Proactive delivery management from the very start is a must. The same approaches that need to be applied to the delivery management of the analysis task need to be applied on a larger scale to the execution exercises.
P-F2	Flexible plans and adaptive behaviour are required particularly during execution, when much will not occur as external plans claimed it would, flexibility and adaptability are essential. Plans should allow for variations in the sequences in which input dependencies occur and for failures of elements of the system and of the integration of elements of the system.
P-F3	Diagnosis and cure activities need to be very flexible so that they can make progress when dependencies are not met or unforeseen dependencies are identified. A set of independent activities that tackle the problem from multiple bottom-up and top-down perspectives should be planned. Each such activity should have alternative methods planned so it is less likely to be blocked by the inevitable disorder and problems that will occur.

Table 1: Principles that influence processes and practices; what is done and how to do it.	
P-F4	Have alternative methods of achieving each technical task. There are always unforseen reasons why something that should work does not when it is applied in anger. If there is no alternative method then you are stuck. So each task or sub-task should have multiple ways of doing it identified in advance.
P-F5	Work out in advance how everything is to be done.
P-F6	Prepare everything that can be prepared. All necessary knowledge, infrastructure and tools must be in place. The only things that anyone should have to wait for are the delivery of the thing to be integrated and the delivery of any changes to it.
P-F7	Everything that could be rehearsed is rehearsed. Rehearsals identify problems that can then be fixed before they impact delivery.
P-F8	Prepare for high intensity front loaded execution. The execution plan should be prepared and the available resources matched to the tasks to be done so as to complete 80% of the integration tasks that become available for execution in 20% of the available time frame and all of them before half the time is expended. Inevitably this will slip but the aggressive plan should help to minimise the fall-out beyond the allocated window.
P-F9	Plan and organise for massively parallel activity. Ensure that it is not the availability of people, logical resources such as accounts nor concrete resources such as equipment or licenses that limits what can be done in parallel.
P-F10	Monitoring and intervention. Implement a system for detecting failing tasks and for applying effective intervention to prevent their failure. Particularly when operating at scale there will be teams with different levels of skill and experience at Integration. Some tasks may prove to be beyond the capabilities of the virtual team allocated to the task and intervention by a more senior team will be necessary to correct this.

These principles provide the framework for effective Integration. The rest of this article describes how these map out onto a staged approach to delivering an Integration cycle and onto practices that ensure the planned Integration work is actually done on-time and to the right standard.

4. Integration; here is an “in a nut shell” guide.

So given the principles as outlined in the previous section how do we go about applying the principles? This section outlines a set of stages that form a coherent approach to delivering Integration. As with any attempt to explain a complex activity that talks of stages the order is nominal and when Integration is being done activities from many stages will be occurring in parallel, out of sequence and iteratively. As with any approach to applying principles it is not the only approach, there will be others, however it is the one that has been developed and documented.

The Participants

Delivery of the Integration work is distributed across the Core Integration Team and a collection of Virtual Teams. The Virtual Teams required for the exercise are identified during the early stages of the cycle; however where there are repeated incremental releases of the same solution some continuity is both a good thing and also likely to be a natural outcome of the analysis process.

The Stages

There are five stages used to describe the activities. Four primary stages focus on the key objectives of achieving Integration. The fifth deals with how we ensure we stay on top of things and avoid degeneration. The stages we choose to identify are introduced in Table 2.

Table 2: A Stage Model for Integration	
Stage-A: Preliminary Assessment	Identification and rating of the integration risks, selection of the ones to tackle and the packaging and allocation of work to teams.
Stage-B: Team Mobilisation and Planning	Selection and indoctrination of team leads. Establishing and bonding teams. Planning of team activities.
Stage-C: Pre-delivery Prevention and Preparation	Activities to prevent the introduction of problems at source. Activities to prepare for activities when the system is available for integration. Rehearsals.
Stage-D: Target Integration	Activities to detect and diagnosis problems. Activities to resolve problems. Confirmation of problems resolutions.
Stage-E: Integration Maintenance and Support	Activities to detect and diagnose degradation. Activities to provide technical support to other teams.

Each of these stages is described in sections that follow this one. The descriptions provide a summary of what needs to be done within each stage and also a light outline of the way to achieve each one. The information provided should be sufficient to enable a reader to shape an Integration campaign.

5. Stage-A: Preliminary Assessment

Hopefully, having read the previous articles and the preceding parts of this article, the reader is aware that Integration is not a trivial endeavour. Thus it should be obvious that Integration has to have a comprehensive plan. Such a plan, at least a decent plan, does not appear without the investment of disciplined effort. Getting the right plan depends upon understanding what needs to be done and allocating the objectives to separate streams or teams who can then plan in detail. Identification and allocation is the remit of Preliminary Assessment.

Approach

The foundation for the plan is established by the core team during the early days of a project. They breakdown the integration challenge. Their findings on integration threats drive the decisions on what is to be tacked, the selection of virtual teams and the allocation of tasks to these teams. It is the core team that deliver Preliminary Assessment.

Objectives

The objectives of Preliminary Assessment are summarised in Table 3.

Table 3: Objectives	
Comprehensive identification of the integration threats.	That is identify those areas where a integration failures could derail the delivery schedule of the programme. Generally this should identify areas where the level of Integration threat is above the background level.
Appropriate rating of the threats.	The threat list should end up as an ordered list. The one at the top should be the most dangerous, the second one the next most dangerous and so on.
Definition of the main cut-off point in the list.	Activities Those above the line must be tackled by Integration; those below the line may not be addressed. The cut-off point needs to be supported by a cost benefit analysis.
Identification of streams, virtual teams and allocation of tasks to them.	Place the threats into coherent groups. Estimate how many teams are needed for each group. Allocate the threats to teams. Identify the teams. Refine the grouping and allocations.

Timing

In the second article timing was discussed with reference to the best time to work out what needs to be done. It was explained that the best time to do this is around the time that the overall programme of work achieves Convergence [reference 3]; the points at which all significant facets of the programme of work come into reasonable alignment.

There are two parts to working out what needs to be done. Preliminary Analysis is the first part whilst the second consists of virtual team mobilisation and planning. At the end of these there is a big list of things to do and a plan to do them. So Preliminary analysis needs to sit at the front end of the time window around Convergence that is allocated for working out what needs to be done.

A recap of the constrains around this window for working out what to do flags the following four points.

- The need for the high level design to exist. Integration is about putting together the bits that the design partitioned the system into and making them work together as intended. The technology portfolio and the architectural design are need to act as a framework for risk identification.
- The need to be near Convergence. A lack of alignment between any two or more significant elements may mean change. Design may change to save time or cut costs or to meet some requirement. There is a risk of wasting effort if work is done where there remains a significant likelihood of change.
- The aim to have all of the identification work, including that dependent on Team Mobilisation and Planning, complete by the time implementation design starts in earnest so that any required intervention in this design activity is known about at the time it is required.
- The need to have preparatory tasks completed by the time components arrive for Integration. Again preparation depends on all of the tasks being identified as a result of this analysis and Team Mobilisation and Planning and also having time to carry out the necessary preparation.

Within this general framework where does Preliminary Analysis get scheduled. It is a relatively high level and low cost tasks; compared with detailed analysis and planning subsequently done for each risk area. It will take time to do properly; allow weeks not days. Given these characteristics the preference is to carry out the analysis once a coherent candidate high level architectural design emerges. This allows some aspects of team mobilisation to be done off the critical path so that when it is determined it is time to start detailed analysis everything is ready. The downside is that if the design has to change then rework is required; but given the need to complete overall tasks identification as close to Convergence as possible this will generally be the right approach.

An aside; full Convergence depends on Preliminary Analysis

From an overall perspective once it has been decided that the Integration discipline is part of the approach then Convergence can only occur when there is a plan for Integration that dovetails with the rest of the facets of the work. This can only be in place once the Preliminary Analysis has been completed and outline resource and schedule estimates have been prepared for the Integration work.

Principal Tasks with Preliminary Analysis

In this Preliminary Assessment period the core team:

- Analyse the system and the proposed system design to identify aspects and elements that create integration risk[1]. Areas where successful integration is a fundamental requirement for coherent operation of the solution are located. They identify a set of Integration-Targets.
- Perform a Delivery Risk Analysis on these targets. They forecast how bad it will be if we loose a day because that element is not integrated. They judge how likely it is that the element will not be integrated when it is first delivered[2]. They estimate how long it may take to solve the problems if it does not work on day-one and business as usual processes are followed[3]. These factors allow a rating of the delivery risks each item represents. A rating of how big an impact on the project progress there is likely to be if nothing extra is done for this item.
- Prioritise the targets using the knowledge gained from the risk analysis. Order them according to the risk rating; but also apply commonsense corrections. The result is an ordered list of the targets to be tackled.
- Put the targets into groups that form coherent packages of work. Draft an initial list of the virtual teams required to deal with these groups. A virtual team may handle a single group or multiple similar groups. In some cases a large group may be split across two or more teams.
- Identify the disciplines and roles required with each virtual team. Select the members of each team. Define the tasks for each team ready for use in the team mobilisation and planning stage. Remember the virtual team needs to include those responsible for specifying, designing and implementing the target as these are the people who can get it working faster than anyone else.

The End Game

At the end of this stage there should exist an identified set of integration risks. These should be based on the contemporary state of the high level design. They will have been weighted based on the severity of the impact on the progress of downstream activities, Streams of work and then virtual teams will have been identified. Some risks areas will have been allocated to teams whilst other, lower risk ones, may have been left to take their course. The members of the virtual teams will have been selected and their task briefings prepared.

6. Stage-B: Team Mobilisation and Team Planning

The second part of establishing an effective Integration plan involves each team planning in detail how they will deliver their remit. A pre-requisite to that is the mobilisation of the teams which encompasses forming the team, assuming it does not already exist , and briefing it on the remit it must deal with.

Approach

[1] Operating with a clear understanding of the distinction between Delivery-Risk and In-Service-Risk.
[2] Previous experience counts an awful lot when making these judgements. If last time we did something like this it was fundamentally broken then it is a good bet it will be again.
[3] Many factors kick in here. How effective are people at identifying the source of the known problem. Are they likely to work serially; encountering one problem then another and so on? What are the constraints on delivering a change; is the team working on other priorities, do we have to wait a week or a fortnight for the next build?

Each team needs to take responsibility for its own work. That said each team also need to apply the doctrine and approaches that permit effective Integration. We must ensure the team understands Integration doctrine and that it develops a plan that will apply this doctrine to achieve the particular objectives the team has been tasked with. This stage is supported by the core team which facilitates and to some extent directs the virtual team up to the point where it becomes an effective self directing element of the collective effort.

Objectives

The objectives of this stage are given in Table 4.

Table 4: Objectives of Team Mobilisation and Planning	
Virtual teams established	The teams exist and have people allocated to them. The members of the team have started operating as an effective team.
Virtual teams tasked	Each team has been fully briefed on the threats it has to deal with. The teams have demonstrated their understanding by reflecting this back in a mission description and presentation.
Virtual team activities planned	Each team has analysed their mission and established a validated plan for delivering that mission.

When is does this have to be done by?

Mobilisation and team planning form part of the “working out what to do” challenge. As such it needs to be done around Convergence and be effectively complete (a) by the time significant amounts of detailed implementation design is starting and (b) in time to allow for complete preparation before items arrive to be integrated. Obviously it commences after Preliminary Analysis; the question is how much if any to delay after this point before it starts in earnest? The actual scheduling needs to balance these factors.

In the worst case it must be effectively complete before the concrete realisation of the system commences. By concrete realisation we mean the production of the XML schemas, database schemas, codes, COTS package configurations etc. etc. that will actually make up the running system. We say worst case because there is a good chance that opportunities for preventative measure during design will have been missed and the window for preparation for delivery will be tight.

Getting the balance right between committing resources prematurely and starting too late is not always easy. The one thing that is certain is that after a number of cycles an organisation will find it easier to spot the optimal points in different kinds of projects within the organisation. If in doubt start earlier and absorb the cost of rework; it will be less costly than a failed Integration exercise.

Mobilisation; forming and tasking the teams.

The first aspect of this stage, Mobilisation, involves forming the team, ensuring the team understands the remit of Integration and tasking the team so they understand their objectives and how they fit into the wider picture. It also means ensuring they have a plan to get going; part a “plan for a plan” and part a plan to learn how to do Integration. The plan is an early deliverable from the team; ideally coming out of or shortly after a kick-off session. The plan covers completion of Team Mobilisation and Team Planning.

For a brand new team Mobilisation involves the full set of activities listed below; for an established team or a team made up of members with previous experience a sub-set of these may be appropriate whilst others may not be necessary or require less focus.

- Each virtual team will need a team lead. They may or may not be the dominant technical lead but they will be the person who ensure things are done effectively on time. The team leads should be taken through the activities that their full team will subsequently go through. This allows them to understand Integration, the role of the virtual teams and their own team's role in the bigger picture. As integration is a collective exercise the team leads should know each other and understand each other's objectives. Common sessions can assist with this and this awareness will contribute to the teams assisting each other.
- The team lead must prepare prior to the kick off session with their team. They need to understand how Integration works, the tasks the team has and the approaches that might prove fruitful. They need to review the proposed membership of their team. They also need to create the outline of the plan of initial activities that will follow the formal kick off for their team; this outline acts as a framework for the team to enrich and refine. The team lead is assisted by a nominated liaison person from the core team who can help the lead to cover the necessary activities quickly and effectively.
- A formal team kick off session is important and must be done well to give the team momentum. It will have a wide agenda including the items listed here. Introduction of the core team liaison lead[1]. Introduction of the team members to one another. Briefing the members on their team's objectives. Explaining the span of their activities and the timeline over which the team will be active. Defining the time-boxed windows for each stage of the team's activities. Ensuring the team has an overview of the remit of other teams. Detailing the short term objectives and pre-scheduled activities for the team; with the key short term objective being to analyse the problem and develop a plan to achieve the overall objective.
- The first deliverable from the team is a work plan to enable them to deliver on their short term objectives and to execute any pre-scheduled activities. This work plan covers the team mobilisation and planning stage; so, when the work plan has been executed there should be a plan to achieve the team's overall objectives. The team lead should have an initial version of this work plan. The final agreed work plan should be the initial team output; it should not be something handed to the team from above.

  As this work plan is formed there will be negotiation with the core team and evolution of the initial input to the team, for example renegotiation of the dates of any pre-scheduled activities, as more detail is uncovered and practicalities, such as conflicting demands, are unearthed. The end result will be a plan that will enable the team to complete the Mobilisation and Planning stage.
- Train the team in the art of Integration. Through initial briefing sessions and then through monitoring of the teams activities and mentoring ensure that as a whole the team understands both what Integration is and what it is not; ensure the team understand how to best go about it. Starting with a briefing from the core team this training may span the history of previous projects and working with other established teams to gain hands on experience.
- Turn the members of the virtual team into a well bonded effective team rather than a set of individuals from separate areas working on the same problem. The members of the team work together on diverse activities to build the team. Some may be essential at this time whilst others may be brought forward or some simply invented to create team building opportunities.
- Engages with the wider world. Become known. Communicate their remit and their approach. Establishes contacts and interfaces with the wider world. A team is not mobilised if it is disconnected from the world around it or if the world around it ignores the team and carries on regardless. The most effective team in the world will fail if it is ignored. The team organises the activities necessary to create the awareness and contacts that will ensure it becomes an integral and respected part of the overall operation.

Team Planning Activities

The second aspect of this stage, Team Planning, builds on the foundation established by the core team to help complete the plan. The overall plan for Integration is the formed from the interaction of plans developed by

[1] This is not the same as any members of the core team who will also form part of this virtual team.

individual teams. The core team broke the task down into separate efforts; the separate teams now define how to deliver each element and build the plan up.

Effective planning requires the team to fully understanding their remit and objectives, to analyse the problem this presents them with and to form a robust plan that will enable them to achieve their objectives[1]. To be successful the team will need to capture and create knowledge[2] about the problem they have, will need to invent good solutions to the problem and then will need to prepare a robust plan to execute these solutions. The team must:

- Engage with the core team to gain a comprehensive understanding of the targets that have been allocated and the objectives that the team is tasked with delivering. Although initially covered as part of the kick off session this will be an extended two way communication activity during the mobilisation and planning period. As questions and concerns are raised and the team's plans form so the remit will need to clarified and potentially refined. The kick of session is the start of this communication. Dialogue and refinement will continue throughout much of the Integration period. The communication must create shared knowledge in the virtual team and the core team of the remit of this team. Common understanding is very important.
- Get up to speed on their domain and the challenges they will face. They share knowledge within the team; they gather knowledge from outside the team. Team members provide briefings to the other team members on the work their organisations have done towards delivering the targets. The sharing and gathering of knowledge and the application of intellectual effort creates more knowledge. Integration analysis techniques are applied by the team to create deeper more focussed knowledge of the problem. Lessons from history can be very beneficial as problems do have an annoying tendency to repeat themselves.
- Work out what they need to do, what they should do and prioritises items based on the forecast benefits. As the team's knowledge develops it will become clearer what the activities are that will provide material benefits. The team will be able to work out a treatment for each target. Treatments cover preventative measures, measures to cure any problems that are not prevented and measure to prepare for administering the cure. The cure element encompasses assessment, diagnosis and fixing.
- Carefully analyse and conceive the cure aspect of each treatment. These activities are the most vulnerable to disruption; victims of the friction and disorder that emerges to cripple progress. Activities can be brought to a halt by an initial problem with the target or by a problem elsewhere. To address this threat the cure is structured as a set of independent activities. Each provides benefit and each is as independent from the others and from the overall state of the target as is possible. Each one eliminates some integration risks reducing the overall risk for this target. Each one should be planned with alternative ways of achieving the same end so that when foreseen or unforeseen dependencies block two options the third can be used.
- Prioritise the treatment measures across all of the elements the team is responsible for based on the benefit they are expected to yield. Based on how much risk reduction they are expected to deliver. The team will not be able to do all of the preventative, preparatory and cure activities within the timeframe with the available resources. Ordering of the candidate items based on the expected benefit is necessary to permit sensible team planning.
- Prepares a realistic delivery plan. Converting what must and should be done into what can be done. This covers that the team will do from now to the end of the exercise. It will show which treatment measures preventative measures, cure measures and preparation measures can actually be delivered by the team. This will not be everything. Where really important things cannot be delivered the team or the core team should consider whether to re-allocate a target to another team. The answer is probably not more resource into this heavily loaded team; teams need focus and this will be dissipated if the team or the task grows too large and diverse.
- Communicate the team's approach and plan to the core team, to other integration teams and to the wider world that the team will interact with. This serves to create wider knowledge and to ensure an aligned effort

[1] As part of this process objectives that cannot be achieved need to be renegotiated.
[2] Knowledge can be created by applying intellectual effort to information and existing knowledge to get new knowledge. In a well bonded team the intellectual effort available is more than the sum of the parts.

across the players involved in the work. It provides an implicit review of the plan with feedback helping to identify weaknesses and opportunities for improvement.

The End Game

At the end of this stage, when all of the teams have established their plans, there should exist a coherent set of Integration tasks that tackle the integration risk associated with the delivery. Backing this intent there should be a delivery plan with a high likelihood of on-time and on-quality delivery. This is the foundation for the Integration of the programme.

7. Stage-C: Pre-delivery; preparation and Prevention

Having identified and planned the work to be done the team has to execute that work. The work associated with an integration target can nominally be split into two parts. Firstly the work that can or must be done before the components of the target are delivered and secondly that work that can only take place once the components are delivered.

This collection of tasks is referred to as Pre-delivery work. It compromises work to prevent problems and work to prepare for dealing effectively with the target and any problems associated with it once the delivery occurs. This section deals with these aspects of Integration.

The importance of the pre-delivery stage.

It is very important to recognise that the best opportunity to achieve integration and to ensure that things become integrated quickly actually lies in the period prior to the delivery of the item for integration. Being effective in this stage is one of the most important success factors. We inevitably find that after delivery time windows are compressed and resources contended. There is more space for effective work on integration before the target is delivered than after it arrives. This opportunity must be exploited to the maximum extent possible. Efforts that do not do so will suffer badly.

> ***WARNING*** *A warning of a common and generally catastrophic mistake. Admittedly there are integration activities that can only be done once the target has been delivered. An example would be executing a transaction that the target supports and obtaining traces of its progress at each stage to spot where it malfunctions. However we often fail to recognise just how much can be done earlier. We assume there is very little that can be done. This massive mistake leads to a lack of preventative measures and a lack of preparedness for diagnosis and cure. It is very important not to get lulled into this mind set; thinking that if we don't have the thing then there is very little we can do. If we do start thinking like that then the chances of success will plummet.*

An investment of effort on Integration can't be left until the target is delivered. We lose days, we are overloaded; we fail! We will not have not done Integration as it is defined here, we have done traditional post delivery fire-fighting and we will get traditionally unpleasant outcomes. Pre-delivery Integration activities are the key to success.

Approach

After that diatribe on why this stage is so important; exactly what can and should be done during this stage? There are two aspects to activity in this stage; these are Preparation and Prevention. Preparation covers getting pre-requisites ready to enable the team to deliver the activities that will prevent, diagnose and cure integration issues. It encompasses both generic preparation of the team and specific preparation for the diagnosis and cure

activities that will be done after delivery. Prevention covers activities that influence and shape the development activities so that there are fewer issues to address when the delivery is made.

Preparation is everything at this stage; well apart from Prevention and that will be covered later. Generic preparation means building the teams knowledge and capabilities so that they can fully understand what they have to do, improve their plans, intervene to prevent issues reaching them, prepare for diagnosis and cure and execute diagnosis and cure. Preparation for diagnosis and cure means getting everything ready for diagnosis and cure before the target is delivered.

Objectives

The high level objectives to be achieved during this stage are listed in Table 5. The main activities that contribute to achieving each activities are discussed in subsequent sections.

Table 5: Objectives for the pre-delivery stage.	
Generic Team Preparation	Common body of domain knowledge established. Teams technology and system competency established.
Diagnosis and Cure Preparation	The team is fully prepared for the start of integration of delivered components. Evaluation work partitioned. Evaluation and diagnosis techniques defined. Evaluation preparation planned. Tools and utilities developed. Techniques and
Prevention	Identified prevention tasks executed.

Generic Team Preparation

Attention and effort needs to be applied to developing knowledge, technical competence, system competence, existing system knowledge and user process skills. The topics covered in this section are:

- The development and refinement of domain knowledge.
- Establishing technology competency.
- Establish system usage competency.
- Familiarisation with existing systems.

Develop and refine knowledge

Continually developing the team's knowledge, sharing it within the team and, where appropriate, with the outside world. A knowledgeable team will be a more effective team. The initial plan will have been built using knowledge developed by the team but knowledge development should not stop there. The ongoing plan should include explicit activities to enhance the knowledge of the team and this must be done well before delivery. Points to note are:

- Knowledge development can exploit background research, workshops, technical reviews, simulation and experimentation. Sources should include critical appraisal of programme deliverables. As the team should contain members who already have relevant knowledge so team members should brief the team on the output from their own organisations. All possible opportunities and methods should be exploited to create and secure more relevant knowledge.

- The variety of knowledge required by the team is wide. It spans the design and how it is intended to work, the way things are built, the order in which things will be delivered, what it is expected to do, how it will be used and why it is required to be like it is. Gaps in any of these domains will reduce effectiveness.

Technology Competency

Hopefully the team will have someone competent in each of the technologies they will need to understand and use; that is one of the principles used when selecting the right team. If so then cross training within the team can ensure all team members achieve the right skill levels. If not possible then use of external training resources may be necessary. Ensuring technology competency should be a formal team activity; there should be a gap analysis and then managed activities that close the gaps. Achieving competency should not simply be left the individual team members.

System Competency

Ensuring that the members of the team are able to use the systems they will have to interact with.

Generally integration will require people to use systems. To be effective team members need to do this; they should not become puppet masters using others by remote control. To be able to do this they need to be competent users of these systems.

The team needs to be able to use both the front doors and back doors of systems. That is they need to be able to use the systems as they are normally used during operation but also to access them and use them in other ways, as developers and support staff would, this allows them to observer how the systems are working and to tamper where necessary.

Once again cross training can be used where the team posses members with the right experience; something it should posses; alternatively external training resources may be required. Again a managed set of activities should be used rather than relying on individuals.

> ***WARNING*** *It is very easy for teams to focus on higher level knowledge and general technology competency whilst forgetting the actually need to be able to drive when the time comes. Driving then turns out to be far harder than anyone thought possible. Driving lessons once a delivery has arrived are very expensive lessons.*

Familiarity with Existing Systems

Where appropriate the team may benefit from familiarity with existing systems; systems that do the same or a similar job to the job to be done by the new system. Experience of existing systems may serve a number of purposes. It may improve the domain knowledge of the team. Alternatively it may be that the existing system act as a reference against which operation of the new system can be compared either at the macro or micro level. Again gaining experience of existing systems should be managed as part of the team's planned preparatory work.

Preparation for Target-Integration; diagnosis and cure

As part of the Preparation stage all preparatory work required for actual Target-Integration should be completed. The team aims to fully prepare themselves, their methods and their tools prior to delivery of the target(s).

Generic Preparation

There are a number of generic areas of preparatory work that any team should address during this period. These are listed here:

- The team should get early visibility; hands-on and competency on the operation of the targets they will need to work on. This may seem contradictory; if the team are going to Integrate the target how can they get hands

on experience before hand. The answer lies in taking advantage of the opportunities where parts of the target will exist and be usable prior to delivery for integration. Examples would be component and supplier test environments; the team should witness component testing, hopefully via the member of the team from that component, and get hands on experience. They should try out the operations they expect to apply during Integration within a component or supplier test environment.

- A particularly important aspect of gaining competency on the systems the team will have to work with is becoming very familiar with the correct way to carry out key transactions and operations on the system. This should become second nature to the team as it will avoid delays and spurious errors due to incorrect use of the system by the team
- Define and walkthrough the methods that will be used for each Integration activity. Having identified a set of activities associated with each target the team now needs to design the working methods that will be used to execute each activity. The activity defines what is to be done; the method defines how it will be done. Methods are defined by working groups; once methods are defined a walkthrough is held with the wider team to communicate the approach, challenge the approach and identify any tooling requirements arising from the method.
- Prepare and prove the working environment. When the target deliveries arrive we do not want to find that there is no environment to deploy them into or that there is but things do not work because the environment does not work. Therefore the environment should be built, if necessary, tested and fixed prior to the arrival of any elements of the target delivery. This should be an carefully planned activity included in overall team plan.
- The team may well need tools, will most probably need tools. Some could be essential, the team would not be able to execute the integration approaches without the tools, others will be needed to accelerate and make more reliable actions the team will need to execute. Tools may range from simple scripts to access log files, scripted database queries and more complex capture scripts across stubs and simulators to comprehensive real-time monitoring and diagnostic packages.
- Tooling requirements emerge from the definitions of methods for Integration actions and from walkthroughs of these methods. In the preparatory phase the team need to procure or build the tools, integrate them into the integration environment and test the tools and the way they will be used. As with anything else the team will have to work with competency in using the tools needs to be established across the team.
- As many of the Integration activities as possible should be rehearsed by the team during the preparatory phase. Now this may be hard as the target has not been delivered but creative thinking is required to give the team as much practice and rehearsal as possible. For example if the team is working on a change to an existing system then rehearsing on the existing system could provide hands on experience and validation, or not, of some elements of the methods being proposed.
- Finally don't forget preparation and rehearsal of administrative tools and processes; for example status reporting and fault recording. If these aren't there, or the team do not understand how they really work and how to engage with them then there will be delay and missed opportunity exactly the things that Integration is supposed to avoid.

Specific Preparation

Less generic requirements for preparation arise form the tasks and techniques chosen during the planning of the Target-Integration stage. More specific requirements can range from procuring and installing a particular piece of equipment to the preparation of a database query to perform a particular task. The reader will be able to foresee the range of some of these task when they have read the section on the Target-Integration stage and become familiar with the sorts of tasks and activities that could need to be undertaken and hence prepared for. Other will only become clear during specific programmes and cycles of Integration. For example organising a daily motorcycle courier to collect items from a physically remote print centre is not going to appear in any generic article on Integration.

Prevention

The aim of prevention is that there should be an absolute minimum number of issues to be dealt when the delivery is made available for integration. This ensures that the work to be done after delivery and hence the time taken to do it is minimised; the key objective of integration. We have two types of measures that work to achieve this, firstly Avoidance and secondly Elimination.

- Avoidance activities aim to stop issues being introduced in the first place. The approach is to focus the attention of the development chain on the potential problems and to shape the methods and tools they use to reduce mistakes.
- Elimination activities aim to remove those issues that were not avoided and aim to achieve this before the item is delivered for integration. They aim to leverage the existing delivery chain to identify and remove the issues that have been introduced within that chain. At this stage most effort is invested by the development teams rather than the integration team.

Prevention: Avoidance Measures

The extent to which an investment of effort to promote additional avoidance measures can help depends upon the nature of the existing mainstream business as usual development processes. Very mature disciplined processes may mean that any contribution may be marginal; immature processes with poor discipline present a vast opportunity for improvements[1]. Avoidance measures are diverse and will vary from organisation to organisation. This section describes some general classes of activity that can be applied to avoid problems but this is not a definitive list and each environment will lend itself to variation and extension of these approaches.

Here we cover:

- Closure of communication gaps.
- Stubbing and simulation policy.
- Extension of virtual teams into the supply side delivery space.
- Targeted design and implementation techniques and tools.
- Integration centric walkthroughs.
- Interface and interaction change management.

Avoidance Measures: Improved Communications

Closing the communication gaps that originate problems is a powerful avoidance technique. Many integration issues arise because teams delivering elements that have to integrate have different understanding of how things are divided and how things are meant to work together. Generally there are three or more parties involved; the party making the decision about how the feature is to be divided and two or more parties responsible for supplying the parts and also for determining the detail on how the parts fit together. Misunderstanding leads to differences which lead to failures. The avoidance action is to ensure clear communication regarding the division into parts and the interaction between the parts. Ensuring the use of disciplined information practices that will reduce the occurrence of errors. The discipline covers information Logistics, Expression and Detail.

- Logistics is about ensuring that all parties are using consistent and correct information sets; it is about identifying the right information and about making that information accessible. Logistics encompasses configuration management as well as distribution. It avoids problems arising from people acting on the wrong information.

[1] Our experience shows that generally there will be significant scope for improvement.

- Expression is about ensuring that knowledge and information is captured in persistent conveyable forms in appropriate chunks; it is about effective documentation. It avoids problems arising from people misreading or misunderstanding the information they have available.
- Detail ensures that information that exists is expressed in sufficient detail. It avoids people introducing integration issues due them having to decide the detail for themselves.

Avoidance Measures: Stubbing and Simulation Policy

Most large scale development activities involve the use of a variety of test harnesses, stubs and simulators. Often more than one component teams will require a stub to emulate the same component. Our experience is that each team writes the stubs they used based on their interpretation of the specifications. There can be multiple items of test code based on the same component. Not only is this inefficient from a development and maintenance point of view with multiple teams doing similar work it also creates integration risk where there are differences between the behaviour of the real component and the test versions.

Harnesses, stubs and simulators need to be a high fidelity emulation of the real component that will emerge. Deficiencies in stubs lead to integration issues. Yet we often have the situation where the team writing a component also produce stubs for the components they interact with. They use the same specifications and the same assumptions for the stubs as they do for their own component. Any mistakes cancel each other out. Their component works fine with the stub but fails to integrate with the real thing.

To avoid these type of issues the production of test harnesses, stubs and simulators should be done by the teams that will provide the real production version of the component. The fidelity of the test tools will be much higher and the value provided by developing and testing against them amplified.

- A general policy should be established that each supplier delivering a component into the wider solution should also be responsible for providing and maintaining:
  - Test harnesses to simulate this components role as clients of other components.
  - Stubs to simulate this components role as a server to requests from other components.
- On a case by case basis the need for the team to provide more sophisticated simulations should form part of the programme planning process.
- Harnesses and stubs should be amenable for use in the creation of higher level simulators.
- Stubs should be amenable to extension of the rules that controls their responses by teams that will use the stubs.

Adopting this approach will not be trivial. Most teams will not naturally become proactive supporters of a policy that requires them to delivery and maintain harnesses and stubs on top of delivering the production components. There will be an argument that this is extra work. The reality is that it is work being relocated, moved to the most effective place it can be executed, rather than being new work. A risk is that the items provided will not support the ways that the other teams intend to use them. Analysis and design must be done to ensure that the items produced are fit for purpose. However despite the challenges even a partial implementation of this approach the spans high risk interfaces will reduce integration risk.

Avoidance Measures: Virtual Teams in the Supply Chain

Virtual teams are the fundamental foundation of Integration. They draw together diverse expertise to work together on a coherent effort. The benefits of virtual teams do not have to stop there. The design and development activity surrounding a significant area of integration threats can mitigate the threat or exacerbate the threats. A collection of independent activities done separately is likely to realise many of the threats; turning them into actual problems. On the other hand a joint activity where all the teams involved work together on a shared design and implementation is likely to avoid many integration pitfalls.

So promotion of shared design and implementation using virtual teams as the mechanism can provide an effective way of avoiding integration problems. Further benefits can arise from aligning the virtual team in the

delivery space with the equivalent teams in the Integration space; often they can contain similar memberships and require similar knowledge.

Avoidance Measures: Targeted Design and Implementation Techniques and Tools

Basic integration issues can be reduced through appropriate use of low level techniques and tools. By ensuring simple techniques and tools that give a high return are being used and used appropriately many problems can be avoided. Sample measures are summarised in Table 6.

Table 6: Improved Design and Implementation Techniques

Technique	Summary
Basic Interface Definition Discipline	Many faults occur at the basic interface level. Data is rejected due to mismatched encoding or unexpected structure. To reduce this risk ensure interface definitions are as rigorous as possible and use tools to validate them where ever possible. • Use, effective maintenance and communication of formal interface definition techniques such as XML schemas and WSDL. Use the most formal notation available. • Use of tools to validate the interface definitions. We see many specifications that are incomplete or contain errors. • Go as far as possible to use of formal restricted enumerations rather than unconstrained string values in interface definitions. We have seen tens of thousands of pounds wasted, possibly hundreds of thousands, because one team sent one thing and the other team looked for something similar (say “Month” compared with “Monthly” or “Restricted” compared with “restricted”). • Apply this where ever two systems exchange information; even if they are not adjacent.
Integration Risk Aware Partitioning	Systems are partitioned into components delivered by different teams or suppliers. In a given area one partitioning can create massive integration risk whist another can greatly reduce this. Design and delivery teams need to be educated as to the impact this can have and persuaded to adopt approaches that reduce risk. This concept can be illustrated by an example. Say a consumer system needs a quasi realtime mirror of an entity’s state where the master is on a source system. Either (a) the boundary can be between the two system or (b) the source system can provide a plugin that is accessed directly on the consumer system. Strategy (a) means that a complex potentially lossy synchronisation protocol has to be agreed and correctly implemented by two teams. Strategy (b) allows this complexity to be dealt with by one team and a simple programmatic query interface presented to the other team. Strategy (a) is high risk and (b) is low risk. What do we tend to see? We tend to see strategy (a) used most frequently. Why? It appears cleaner and easier to support, modern doctrine indicates that the boundary of responsibility sits on web-service and XML interfaces and it is how it is generally done. • Start a discussion or debate about the way partitioning can affect risk and alternative approaches. • Identify high risk complex interfaces that can be replaced by simpler interfaces if an alternative partitioning approach is adopted. • Campaign to have these alternative partitioning strategies adopted.

Table 6: Improved Design and Implementation Techniques	
Technique	Summary
	• Where this fails ensure a virtual delivery team is assembled to work on the high risk interface.
Information Mapping Discipline	Modern methods, particularly on large scale distributed architectures loose sight of the overview of information, what it represents and how it is structured. As a result in large systems many distributed capabilities and behaviours fail to work because of information incompatibilities. There can be different conceptual interpretations of information that distributes across the solution. There can be different expectations of how complex information will be normalised and represented, These are not low level data exchange syntax issues that can be addressed using interface specifications. These are higher level information modelling issues. Normalisation and representation issues are the lower level problems here. If a set of addresses contains the addresses that a set of people live in then what happens if two people live at the same address? Is the address present once or twice? If a person has no middle name is a field dropped or sent blank. Are leading and trailing spaces removed from strings. Are dates sent in US or UK format. Conceptual issues are at a higher level. In a CRM system what does a contact represent? Is it a person; the aim is to always have one contact for each person in the real world we are involved with? Alternatively is it a communication channel with a person of which there can be more than one or even a communication event with that person of which there could be many? Traditional technical descriptions do not address this; an entity model may say what a contact is and what it links with but not how it is really intended to be used. To address this sponsor the use of a consolidated data dictionary; very old school but very useful, for global aspects and aspects that are not solely related to an individual interface. Also use additional semantic explanation within the interface definition for explanation is only applicable to that interface. • Establish a concept centric catalogue and then explain how the concept manifests itself in each system. So if we have a concept of a person it could turn out that it appears in the CRM system as a contact but that there is no equivalent in the web portal as we a expect individuals to have multiple web accounts and so a web-account is not a person. • For each separate representation of some conceptual information that is transmitted between systems record how real world data is normalised. How do we deal with the more complex cases. • For each interchange ensure that the more semantic and information dependent aspects of the data that will be presented are documented. Ensure this is done in a generic way and does not depend on examples. Examples are good but they don't cover the cases that are not turned into example.

Avoidance Measures: Integration Centric Reviews

Reviews are most effective when they are focussed on particular types of problems. Integration can take advantage of this by organising reviews that focus on the integration threats within the system. This is not necessarily a review of a single artefact, for example a design, rather it is a review of how things will work that uses the body of knowledge and artefacts that have been emerged from the design process. One useful technique is to get each team to describe what the other system does rather than what their own system does as this will promote engaged discussion of the topic.

An example topic could be a review of the operation of the mechanisms that maps and transfers customer information updated in the online registration portal into CRM system and reflects CRM updates back to the portal. The review could cover:

- A general description of how this integration of information storage and updates will operate from the implementation team with the CRM team explaining what the portal does and the portal team explaining what the CRM system does.
- Discussion of the information content exchanged and how it is used in the receiving system.
- Presentation of information encoding and key features of the encoding.
- Explanation of information exchange mechanisms and how they will deal with anomalies.
- Consideration of how time delays would impact the behaviour.
- Consideration of the impact of operation when parts of the system are out of service and of recovery to a healthy state after service is restored.
- A walkthrough of a set of operational scenarios to obtain from the teams implementing each component the story of how their component will behave as the scenario progresses.

This illustration was just that; a single example of how one vulnerable area could be reviewed to flag issues early. Each area for review will requires an agenda like this and an approach to orchestrating an effective review. Each one will be different and each one will require careful planning by the team handling that area.

Avoidance Measures: Interface and interaction change management

For established systems one of the major sources of issues are changes that have unforseen effects. Changes that break things that worked before. The underlying issue are changes that takeaway something or change something that other things depend upon. The solution is either to coordinate changes so everyone adapts to the new behaviour or to ensure that when we create a new one we also maintain backward compatibility and leave the old one in place as well.

This does not happen by magic and on large scale systems it can not happen informally. On large scale system there needs to be an active process that identifies and manages interfaces and interactions between components. Characteristics of this approach are:

- Identification and maintenance of an inventory of interfaces and of dependencies on the interfaces.
- Identification and maintenance of an inventory of interactions between components and behaviour distributed across components.
- A process where by any design or proposed component changes that would affect any of these elements are reviewed to assess the wider impact and any requirement to maintain backward compatibility.
- Effective assurance processes to ensure that the results of this analysis are applied effectively/

Prevention: Elimination Measures

Elimination measures aim to use the mainstream development testing and assurance methods to detect, diagnose and eliminate integration issues that have got past avoidance measures. The aim is to have them detected and removed before the date that integration of that area is due to start. It is part of the world view that says that the supply chain as a whole should be delivering a solution that is integrated rather than a collection of bits with integration problems. Note that if the combined avoidance and elimination measures are highly effective then there will be little detection, diagnosis and cure to be done and deliveries will rapidly be flagged as integrated and ready for wider use.

Here we cover:

- Static validation of messages..
- Next Box and Peer Box testing.
- Continuous Integration testing

Elimination Measures: Static validation of messages

A recurring class of failure is systems emitting messages or making calls that can not be parsed by the receiving systems. This occurs despite the existence of proper interface specifications in the forms of WSDLs and XML schemas. A big reduction in these sorts of problems can be achieved by insisting that tools are used to validate the messages produced during component testing against the formal definition of the interfaces. This task is an organisational challenge than it is a technical one. Features of this approach are:

- A controlled set of interface definitions to validate messages against.
- An inventory of messages to be produced by each component.
- An inventory of scenarios to be addressed for each component.
- A derived inventory of valid combinations of message and scenarios to be covered.
- A mechanism for message versions to be supplied for validation with a history, coverage measures and results.

Further extension of these techniques can include the creation of more restrictive derived specifications for particular scenarios to reflect semantic constraints not directly represented in the interface specifications; for example you can have a house name or a house number but not both.

Elimination Measures: Next Box and Peer Box testing

These terms are used to refer to cooperative testing carried out by the supplier teams to find integration faults independently and hopefully before any integration and testing of the assembly containing these components. The teams are required to self organise to carry out effective testing to demonstrate and validate that their components work together. This becomes part of their normal development processes.

Next Box is generally used to refer to testing between components with direct interfaces. For example if component A makes a web-service call to component B then they have a next box relationship. Here the component teams would be expected to set up connectivity between their own test environments using representative transport mechanisms to demonstrate that these exchanges work. The rest of the system may be omitted or stubbed but this tests real exchanges between real versions of these components.

Peer Box addresses situations where the components are not adjacent but where there is a strong coupling. Peer box can be illustrated by an example. A website prepares a complex XML object representing an order; this is then be transferred, unchanged, via a number of other components until it is finally processed by an automated warehouse dispatch system. The risk is that the warehouse system can not process or does not correctly interpret the data sent from the web site. Next box testing would not address this risk. Peer box testing involves the teams establishing a way, not necessarily via the normal route, of exchanging these messages so their

correct generation, parsing and interpretation can be tested to provide the same benefits that next box testing provides for adjacent components.

Elimination Measures: Continuous Integration Testing

Continuous integration refers the recurring execution of a set of tests that measure how integrated the system is. These tests are at the end of the delivery supply chain. The system is built and then these test are executed to evaluate it. Automation of these tests is essential and this can be complex in large scale systems. From the point of Integration these Continuous integration tests are a tool to prevent degradation in integration. They show if something that was working during the last cycle of testing has suddenly stopped working.

Continuous integration testing, whilst not a direct contribution to accelerating the first off integration of any particular feature, is a strong supporting element. It helps to maintain the integrity of things that have already been integrated thus avoiding both the need for rework and the risk of latent failure creeping into completed items and then undermining subsequent efforts.

Characteristics to be looked for in a continuous integration process are:

- A repeatable, rapid and robust test set that exercises interfaces and other integration risks. Not only must the set exercise the risk it must have sensitive failure detection. Best practice would have an automated test suite to provide repeatability and speed and continuous operation with a reasonable resource cost.
- A balanced model to control the start of a test cycle and if necessary abort and restart of a new test cycle. In the ideal world Agile world builds are continuous, a build occurs on each check in, and each build is tested. In the world of large scale systems engineering a less idealised pattern may be required. At a large scale builds may take a significant time, deployments for test may take hours not minutes and may involve manual deployment activity and a meaningful test cycle may take a number of days. The idealised model needs to be tempered and a pragmatic approach to optimising the error detection effectiveness established.

The End Game

Whereas the previous stages had quite distinct end states and all streams could be expected to conclude around the same time the end game for this stage is less clean. Different teams will move from prevention to actual integration on their targets at different times and the switch for a single team may be staggered. Re-work and change may mean more preventative action; it is the focus that shifts rather than there being a hard stop.

What denotes that the work that has been done is the correct work and that little if anything was missed. It is the effective, and there is an emphasis on effective, discharging of the preventative tasks that were identified as required to bring the integration risk down to an acceptable level. It is the effective preparation of the team, their techniques and their tools ready for the work to be done on the system. Whether or not they were done effectively will be shown during the next stage when the collections of components arrive to actually be integrated.

8. Stage-D: Target Integration

When components that are supposed to go together to implement part of the solution start to arrive then so the Integration tasks that deal directly with the evaluation of each target commence. Components are deployed into a joined up integration environment that has already been proven as part of preparation and are exercised to see if they are integrated. Where problems are found the work expands from just evaluation to include diagnosis and cure. It is the stage that deals with the issues that were not avoided and that bypassed elimination.

At this time the teams must face up to and overcome Challenge four - Executing on time and effectively in a hostile environment [reference 2]. Complex technical activities, ones that check, debug, work around and fix integration issues, have to be delivered in the face of fundamental mismatches, missing bits, system instability and confusion. Avoiding getting bogged down and blocked is a key to success.

The Approach

A virtual team provides the knowledge and skills together with hooks to get support and action from the wider organisation. Detailed understanding of the areas to be integrated built up over previous stages is combined with a set of rehearsed methods for tacking the task. The plan allows for detailed checking individual parts of the integrated item from the bottom up and for top-down demonstration of overall operation. Each of these tasks is as independent as possible; avoiding dependency on other tasks having been done and wherever possible on other things actually working. There are multiple ways of doing each task and checking each item. Work starts as early as possible; it will begin even if there is only a partial delivery. The team's plan aims perform work and make progress in the early part of their declared window so that they can have nothing to do in the later parts. There is a constant review of what to do next and how to do things differently in the light of progress so far.

Integration aims to identify and address inter component issues that would block or significantly disrupt downstream activities. This can range from cases where things simply do not work to cases where something generally works but is sufficiently inconsistent to be disruptive to downstream activities.

Objectives

The principal objectives against which the performance of Integration in this stage can be judged are catalogued in table 7. Activities that posses these characteristics will accelerate the rest of the programme and make outcomes more predictable. Ones that lose some of these will make less of a contribution

Table 7: Objectives for Target Integration	
Day One Feedback	Feedback on the state of each delivery is provided within the first day. Any issues identified are the ones with the biggest impact. Subsequent output identifies more detailed lower severity issues.
Rapid Assessment Completion	Comprehensive assessment of the state of integration completes within days not weeks of the target becoming available.
Rapid Resolution of Blocking Issues	Blocking issues, those that have an impact on the ability of others to progress their work, are resolved either by a permanent fix or an effective and appropriate workaround, within days not weeks of the target becoming
No Leakage	When the Integration team declare that they have completed their work on a particular area all issues that fall under the banner of integration issues have been identified.
Resilient Activities	Activities progress delivering material progress with the solution becoming more integrated despite missing or delayed deliveries, parts of the solution not functioning in an integrated manner or flaws in the way it was planned to tackle.
Parallel not Sequential Checking	Checking of items does not depend on other items either ones earlier or later in the processing chain working or even being available.
Comprehensive Checking	Within the remit of Integration checks are comprehensive. Detailed operation of each mechanism is checked even if higher level functions that use it appear to work.
Enhance the Continuous Integration Suite	To protect the investment of its time and effort the Integration team should ensure effective tests of each item it has integrated are built into the Continuous Integration Suite.

Evaluation and Issue Detection

Evaluation and Issue Detection covers the work done to see if the components do work together sufficiently well for that aspect of the system to be used in downstream activities. These activities either confirm that things are operating correctly or identify the faults and issues that need to be addressed.

The topics covered in this section are:

- The use of predefined assessment criteria.
- Coverage of the different layers of a target rather than just the surface layer.
- Decoupling of tasks and methods to avoid blockages.
- The essential use of invasive methods.
- The use of disruptive methods.
- Using diversity to cope with oversights and other problems.
- Exit criteria for evaluation and issue detection.

Integration Assessment Criteria

To ensure testing and assessment done from an integration perspective is appropriate and that it is done in the most appropriate order it should have been planned around and its execution should be guided by the Integration Assessment Criteria developed for the item during the preparation stage. To recap these criteria:

- List the characteristics that constitute the element operating in an integrated way.
- List the plausible integration failures.
- List scenarios that create particular threats to the correct integrated operation.
- Lists a derived set of combined characteristics, failure and scenarios.
- Identifies which of these have been transferred outside of the Integration domain and are to be addressed elsewhere.
- Prioritises the remainder; so that if there is any competition for resource then it is clear what gets first call.

Layered Focus

In any complex system the integration targets will also be complex. Within the overall function to be integrated there will be sub-functions that need to integrate and mechanisms that facilitate this. The approach to testing and evaluating the integration needs to reflect this layering. There is a common tendency to focus all testing on the top layer. If testing is solely derived in from this layer then the risk is high that significant problems occurring in the lower layers will be missed. Just because the vehicle is still moving does not mean that the engine is not about to overheat. Similarly, but less likely, testing could all focus on the lower level elements and miss issues in layers higher up. Effective testing requires that each layer needs tackling with each element in a layer being addressed.

The concept of layering can be illustrated with an example:

Data Replication System

For example a capability that replicates user information between a master store and a slave store may have a push function to move data at the time of change, a deferred push function allowing for the slave being offline at the time of change and a separate pull function for bringing new slaves into the system. Within these there may be mechanisms to provide reliable message transfers between the master and a slave and error recovery mechanisms that deal with message failures. These may be supported by web-services that permit slaves to register and notify their state. All of this may be built on top of a multicast secure message transport service and a secure robust load balanced HTTP web-service transport solution.

For this example integration testing may choose to handle separately:

- The HTTP web-service transport solution. Does it work at all? How does it behave when responses are slow? What does it do when a network glitch drops a connection? Does it leak resources and degrade over time?
- The multicast message transport mechanism. This would require a similar approach to the HTTP web-service transportation sub-system.
- The slave registration and state management service; web-service by web-service. Do calls get through; does the connection get accepted, does the request pass authentication is the XML encoding of the call parsed correctly? Are field values accepted and are they mapped to the right meanings? Likewise do replies work? What happens when error responses are sent? What happens if no reply is received; does the retry mechanism work?
- The slave registration and state management service. Are registrations successful? Does the slave state on the master track the state sent from the slave? What about start-up?
- The core change push service. Are data changes transferred correctly to the slave? Is it consistent for rapidly changing data? Is it consistent when there are multiple slaves? Is it consistent when some slaves are down? Is it consistent when a slave is not acknowledging? What happens if an update value does not arrive at a destination?
- The deferred push service. Is the correct data sent? Does it overwrite later changes that were sent before the deferred data? What happens if multiple slaves needing deferred writes come back on line separately?
- The pull service. Does it work? Is the correct set of data pulled? What happens if the delivery is interrupted and has to restart? What happens if data is changing during the pull?

Each of the listed areas is a separate concern that has to be checked out by tests looking for integration issues. The correct integrated operation of each part within each layer needs to be confirmed with an appropriate level of rigour. This is the layered approach to integration activities.

Decoupling

Within any mechanism of any layer actions tend to operate in chain. Once again using example ? on page ? as the basis for illustration:

- In the web-service layer there is a chain where by (a) a web-service call is made from the slave to the master and (b) a reply is sent from the master to the slave.
- In the web-service layer there is a chain where by (a) a web-service call is attempted to the master, then (b) no reply is received and the call times out and then (c) a re-try occurs and the call is made again.
- At the top level (a) a slave registers, then (b) having registered it executes a pull data operation to get its data synchronised and then (c) it starts to receive push data.

Generally normal testing will treat chains as atomic test is planned to start at the first activity in the chain and flow through to the end. So a slave registers, pulls the data across and starts receiving updates. A slave receiving updates is taken offline, waits a period and then goes online at which point it is sent deferred updates. Along the way this activities test the normal operation of the web-service calls.

This approach is susceptible to being blocked by point failures. Point failures will take time to fix and the overall exercise will stall. For example if the slave registration XML payload is not acceptable to the master web-service API then all bets are off and nothing can proceed.

Decoupling is a fundamental to the success of Integration. It is the only way to overcome the threat from point failures. In this approach separate standalone methods are prepared to test each separate activity on the chain. This is not just for chains in the top layer it is done for the chains within each layer. The essential feature is that method is does not depend upon any predecessor or successor elements of the chain working or even being attempted.

Again revisiting our example to provide an illustration of this approach could see techniques developed to:

- Create slave registrations directly on the master.
- Establish an initial data set in the slave.
- Force the master to treat a slave as going offline or to bring it online.
- Intercept a message in-flight to allow incompatibilities to be suppress

With this sort of tool box available the dependencies on other parts of the chain can be broken or significantly relaxed. Checking out a activity no longer needs the steps before it to work. As importantly it no longer has to its turn to be checked after its predecessors; they can be checked in parallel or it can be checked before them if it is higher risk and resources are limited. So now the approach could see:

- A slave created directly on the master and then directly given an initial data set allowing immediate checking of the master to slave data push capability. The late delivery of the slave registration service does not stop this important activity.
- Creation of a workaround that allows the correct handling of slave registration requests within the master to be confirmed. It is found that the registration requests are rejected during XML parsing because a tag that was specified as optional is being treated as mandatory. The XML message is intercepted on the wire and the tag inserted to allow the message to be processed and the payload handling to be assessed. This reveals another issue in the mapping of the slaves identity to a message queue address. Without the workaround this issue would not be found until a fix for the tag issue arrives in the next weekly release.

Invasive Methods

Effective Integration depends upon the use of invasive methods. Use of standard user interfaces and external APIs is not an adequate approach. With only these interfaces to play with it is not possible to decouple the activities in a chain. It is not possible to take control and simulate atypical scenarios such as internal anomalies and timeouts. The visibility of the way the system is operating internally is poor. It is not possible to confirm whether or not things are working as intended.

Returning to an earlier analogy. If the test driver is sat in the driving seat of the vehicle on a short test drive and the vehicle is moving along then there is no way that they can tell that the engine has a slow coolant leak and that the alternator is not charging the battery because of a bade connection. Similarly the driver is not in a position to inject test the effect of a transitory bad connection on an ignition lead. To do all of these you have to be under the bonnet or hood; depending upon where you are from.

Integration has to use invasive techniques both for testing and for diagnosis. These can include:

- Direct interaction with databases that are not intended to be accessed other than by the executing software. Inspection of the contents of the database is used to verify its presence and integrity or perhaps the absence of data after deletion. Direct manipulation of data is used to simulate the actions of earlier activities or to correct the data to allow subsequent activities to commence.

- As a matter of course all messages and interactions flowing between systems should be intercepted and captured. All items should be accounted for. The set should be validated for completeness and for correct sequencing. Spurious unexpected items should be identified and addressed. Messages should be validated against interface specifications and scenario rules and payload values inspected.
- The mechanisms supporting the operation may be monitored to reveal hidden malfunctions. Queue lengths, available connections and other indicators are monitored. Log files are filtered to highlight unexpected warnings and error messages.
- Interception and manipulation of messages to work around problems or simulate conditions that can not be directly triggered.
- Intrusive observation as the norm. For any system activity generated by the team there should be mechanisms in place as a matter of course to capture all intermediate messages together with log files and other indicators associated with processing activity. This information should be smoothly collated so that it can be reviewed effectively.

These invasive measures require technical knowledge and skills combined with methods and tools developed and rehearsed well in advance of their application.

Disruptive Methods

The intrusive methods described above are also used as disruptive methods to ensure the integration is not too vulnerable to operational anomalies causing it to be too fragile for use in downstream activities. Anomalies inevitably happen. In a test environment not they are more prevalent than live; not only are all the normal source present but we now have operational errors caused by bugs that have yet to be fixed and more frequent changes in the state of the environment. So being integrated so long as everything works perfectly well is not good enough, to be effectively integrated the element has to be reasonably robust to events it will regularly encounter.

The Integration team need to exercise the system with disruptive tests that simulate these problems and address any areas where the integration fails as a result. This is important because of the expense, both time and resources, that will be incurred if large amounts of downstream effort is applied to a system that is too fragile.

The nature of this facet of the integration testing must derive from the findings of the threats analysis performed for the area. It could include many things; some examples are:

- Non arrival of request messages or calls at the server system.
- Loss of replies sent from the server.
- Requests delayed until timeouts have expired.
- Replies delayed until after timeouts have expired.
- Corrupt messages, calls and replies that are not recognisable to the receiver.
- Recognisable and acceptable messages with invalid payloads.
- Out of sequence message arrivals.
- Messages relating to non existent transactions.
- Crashes and communications losses, both transitory and persistent, part way through multi-stage interactions.

Diverse Approach

Diversity is a key feature of the Integration ethos. It is no good having just one way of doing something that depends on a certain screen being available. When that screen is not delivered then the activities that depend upon it are blocked.

Each method should have a number of ways of performing each step in the methods. Each task should have more than one method that can deliver the tasks. By planning in diverse approaches and rehearsing them the team avoids the need to improvise ad-hoc methods of working when time is very short and everyone is under immense pressure because things are already running late.

Diversity can take various forms:

- Having multiple ways of triggering the operation of a particular interaction spanning different parts of the system. For example:
  - As the normal consequence of some preceding activity.
  - By direct manipulation of the contents of a database to cause an action.
  - Running a custom command line tool to trigger the execution of a method.
- Being able establish whether data has reached a destination system both by using functions on the system if they are available and by a direct extract and comparison with source data should they not be.
- Identifying multiple transactions that use a particular service and understanding how these show if the service has operated correctly so that at least once can be found to exercise it.

Exit Criteria; when to stop?

Evaluation and issue identification stops when there is confidence that there is a low probability that there are more integration issues to be identified. This does not mean that there will be no more issues nor that there is not a belief that more issues will be found; what it does mean is that it is not expected that any that will be found will be integration issues with a significant impact on downstream activities. From an Integration perspective all that can be discovered at this time has been discovered,

What constitutes this state? Obviously the planned set of evaluation tasks has been discharged but also as weaknesses are spotted that were not originally thought of the set of tasks has evolved to address these areas. Confidence exists not because correct operation has been occurred on a single occasion but because operation has been observed on multiple occasions under varied conditions and because comprehensive assessments have been made of the operation in each case.

The team can give an item a clean bill of health when:

- They have executed their planned evaluation tasks and covered their predefined evaluation criteria,
- They have addressed any further risks identified after the target was delivered as thoroughly as the ones identified prior to delivery during the planning stage.
- They are confident behaviour is consistent under the operating conditions that are likely to occur when executing downstream activities.

Diagnosis & Characterisation

Once a potential issues is spotted it is subject to diagnosis and characterisation by the team. Diagnosis involves identifying the cause the issue. Diagnosis means identifying the discrepancies and omissions that are leading to the failure. Characterisation involves working out how much of a problem there is; when it occurs and when it does not. Characterisation bounds the issue. The two elements together provide the foundation for prioritisation of and delivery of subsequent work on the issue.

The topics covered in this section are:

- Challenges faced during diagnosis.
- Pre-requisites for effective diagnosis.
- Characterisation of issues.

Diagnosis Challenges

Effective diagnosis of a problem requires the team to work in 'debug mode'. Operations have to analysed step by step identifying the points where it first diverges and then potentially diverges further from the intended operation. Diagnosis depends on the ability to observe in detail what is happening and to be able to break and pause the activity at arbitrary points. As with any debugging of a system final understanding comes from looking at both the sequence of things and the state of data in detail. Clear visibility of this information is essential. Speed is of the essence and so visibility and control must be directly available to the team members.

Multiple runs will be required to fully understand the problem. Some runs will be disrupted by outside factors requiring more runs. To avoid delays the team has to be geared up to repeat the activity numerous times to eventually provide the knowledge they need to identify the cause or causes.

Diagnosis can be particularly difficult if the cause involves timing or to some apparently unrelated factor. This issues can make the failure appear random; though this random appearance does not necessarily equate to infrequent.

Effective Diagnosis

Effective diagnosis is dependent on the team's detailed understanding of how it is intended to work. It is also highly dependent on the teams ability and readiness to be invasive. It depends on them having visibility of all of the low level of the system's operation, on them being able to break and pause where necessary and it depends on their ability to intervene and amend the operation to see what works and what doesn't. Major factors in the success of this are:

- The team having a comprehensive knowledge of how the item is intended to operate.
- The team has worked out in advance exactly how to observe the operation of the item in very great detail. They have ways of observing the sequence, timing and data characteristics of the operation.
- They have also worked out how to intervene. Triggering operations at intermediate points and intercepting to delay or amend an exchange are all within their capabilities.
- Not only have methods been thought through in theory but effective ways of putting them into practice, ones that are not time consuming or error prone, have been devised and proved in advance.
- On the back of this an effective debugging environment exists or can be brought into existence with very little delay.
- Tactics for diagnosing issues with different characteristics should have been prepared in advance; these should be used as the basis for tackling individual issues.
- Each in-flight issue requiring diagnosis should have a lead allocated from with the team.
- Under the supervision of the lead a diagnosis road map should be drawn up. This should attempt to relate significant information about the issue (known and unknown) to conclusions about the cause[1].
- The team should then selecting the most effective sequence in which to carry out debug runs in order to converge on a solution to the problem.

[1] This will only be partial as "you don't know what you don't know"; however an initial partially complete map is better than just randomly trying things.

- The debugging and analysis can then commence following the selected order as best it can. Throughout the exercise as more information is obtained the diagnosis plan is adjusted to reflect the implications of this additional knowledge.

Characterisation

Whilst diagnosis is focussing on the known examples of the problem characterisation takes one step back to work out how significant the problem really is. It is all too easy to misjudge this from one example. Misjudgement can go either way. Having been seen once something that actually only happens under very narrow conditions or randomly once in three hundred attempts can be come the focus of a team. The reverse can also happen, something can be dismissed as a one off because it was seen under unusual circumstances when in reality it or a very close relative will be widespread an damaging. Characterisation aims to prevent these mistakes.

Characterisation uses various checks to ensure there is a clear understanding of the extent of the issues. Approaches to be used include:

- Repeatability check; testing whether when the same scenario consistently exhibits the same problem. If this is not the case then determining on what percentage of occasions the problem does appear.
- Divergence checks; starting with the known failure scenario varying some aspect of the scenario to attempt to find the scenarios closest to the failing ones that do not exhibit the issue.
- Convergence checks; started with similar scenarios that are known to be free of the issue vary aspects of the scenario to identify when the problem starts to appear.

The findings of divergence and convergence checks each inform the next most appropriate checks to be made. The information from the two is integrated to provide the overall picture of when the problem occurs, or is more prevalent, and when it is not.

Once characterised the impact on downstream activities can be assessed and on this basis the priority for resolution of the issue can be assigned. If the effects can be tolerated then it may be deferred into the normal fix process rather than being managed as a primary integration issue.

Workaround and Correction

When a problem is understood and it is not deferred then there are two aspects to its further treatment within primary integration:

- Provision of an effective workaround.
- Correction.

Whether or not a work around is required will mainly depend on how quickly the problem can be corrected once and for all. The less severe the problem the longer it can be left without either a correction or a workaround. Problems which block or disrupt significant amounts downstream activities are likely to need a workaround. The team is responsible for identifying suitable workaround and if necessary industrialising it so that it can be brought into general use.

Correction depends upon delivery of changes that remove the problem once and for all. Given their expertise in the area the team needs to work with the design and development community to expedite the identification and delivery of a comprehensive solution. When the solution is delivered the team must validate that it is indeed as comprehensive as expected.

Provision of Workaround

The requirement is to provide a suitable workaround to the teams and for use in the downstream activities that are blocked or disrupted to such an extent that the overall progress of the project is at risk. This does not mean everyone that is affected gets a workaround that suits their need. Nor does it mean that everyone gets the same workaround. Each impact has to be dealt with on a case by case basis.

A workaround does not only have to be technically possible it has to be practical and suitable for application by the intended user base. Ensuring it is practical and suitable means addressing issues like:

- The amount of time using the workaround adds to the task; particularly where it is to be used repeatedly,
- Complexity and likelihood of human error in its execution.
- The magnitude of the logistical challenge in setting up to use the workaround; for example the number of login accounts and tools required.
- The level of technical understanding required to execute the workaround.
- Sensitivity to external factors such as delays between stages of the workaround or the presence other activities in the system at the time.

The aim should be to simplify the user demands of the tasks either by design of the task itself, by automating simple but error prone steps and sequences and by providing clear illustrations of its use.

Workaround can take many forms a list is provide to illustrate this but no comprehensive list can be provided.

- Avoidance of particular data ranges that trigger a problem.
- Use of particular options that prevent the problem.
- Splitting an interaction into a number of separate interactions to prevent the problem.
- Logging out and logging back in to bypass a problem.
- Extension of timeouts.
- Disabling of automatic consumers on a particular queue to allow message editing prior to consumption.

Correction

The role of the Integration team in the correction of an issues is four fold:

- To expedite the process to provide a correction as soon as possible.
- To assurance, in so far as is possible, that the chosen correction is correct and appropriate before it is implemented.
- To ensure, in so far as is possible, that the implementation of the correction is correct.
- Once it is delivered to validate the correction to check whether it is effective and if so to identify and capture any residual issues or further testing required.

The emphasis is on getting the most appropriate correction and getting it right first time. Having to fix the same area[1] more than once wastes schedule time and money.

It is worth observing that the Integration team may actually contain some if not all of the people responsible for fixing the problem. Remember this should be a virtual team drawn from the different parts of the organisation involved in the delivery. If so the team can identify and schedule for delivery the fix; acting as the foundry from which fixes emerge. Alternatively it may have links into each of the teams involved through more junior representation in the team; in this case coordination, sponsorship and review may be the key activities.

Effective correction depends upon effective Diagnosis and Characterisation. These provide the basis for assuring the proposed fix(es) is(are) the right one(s) and for validating the fix(es).

Validation of each fix is very important; particularly if the problem was a complex one to solve or sits within a complex function. The best thought through solutions miss things; things that come to light when the solution is delivered. As well as assessing whether the main problem has been fixed the validation process should identify

[1] Note ‘area’ and not ‘issue’. It is a poor situation if multiple issues of a similar nature have to be fixed separately. It is also poor if fixing an issue introduced further issues. A problem area should come under scrutiny from more senior personnel because it has been found to be a problem area. This should ensure the area is comprehensively reviewed and addressed. Hence each area should only require one fix cycle.

any less important residual issues, characterise them and capture their details as new issues. Once this has been completed the issue can be treated as resolved.

The End Game

As with the previous stage there is no single point where all streams of work simultaneously complete. It is a ragged boundary; made so by the nature of the work being undertaken. For each individual stream the end game is one where the solution is broadly integrated and with enough integrity to allow downstream activities to move apace.

9. Stage-E: Integration Maintenance and Support

The final stage of Integration comes after an element has been integrated. Actually this is not so much a stage that follows on as service wrap provided around items elements that are declared as integrated and that others now depend upon. The wrap has two aims; firstly to support teams undertaking downstream activities who may be less technically skilled or have less specialist knowledge of this area and secondly to ensure it doesn't get broken and if it does to know before anyone else and to fix it quickly.

Approach

The team that integrated the element retain responsibility for it. They assist when it creates problems for downstream activities. They determine whether it is or is not the cause of problems manifesting themselves within the operation of the system. The maintain an ongoing check on its health and tend to it when it fails.

Objectives

The key objectives are listed in Table 8.

Table 8: Objectives for the Support and Maintenance Stage	
Maintain Integration	To ensure that the integration of the individual element is maintained. The high volume of changes that occur as issues are identified and fixed and as additional change requests are implemented inevitably creates a high risk of
Support Downstream Activities	The provision of technical support to teams involved in downstream activities; clarifying their understanding of the way the integration works, assisting them executing activities that depend on integration and helping with diagnosis where it is suspected that integration failures are the cause or when all other avenues have been exhausted.

Integration Maintenance

The amount of resource consumed by maintenance needs to be kept to a minimum. This means limiting it to (a) pre-delivery activities where changes are identified as creating a risk to one or more of the items that have been integrated and (b) responding to breakages that occur. It means avoiding placing demands on the team's time to undertake repeated testing whenever there is a delivery whilst still aiming for immediate detection of any breakage.

Given these constraints the approach to maintenance has aim to utilise ways of detecting integration failures that do not involve the team executing tests each time there is a change. A model that supports this objective builds on combinations of:

- Integration centric checks forming part of any continuous automated build testing. For example validation of messages generated in build tests against target interface specifications.
- Automated next and peer box integration tests applied to the latest qualifying build.
- Automated intrusive integration maintenance tests applied on build deployment into a test environment. If possible into an integration qualification environment that is separate from the environment(s) used for primary integration and for activities downstream of primary integration.
- Operational monitoring and auditing using deliverable diagnostic tools and/or integration specific diagnostic tools to observe integration failures from observation of operations arising from downstream use.

The team must put in place an “early warning system” that provides near instantaneous detection of integration failures and allows them to respond rapidly to any issues. Designing and establishing a mechanism to do this is not something to be left until it is needed. Preliminary design and initial implementation is a preparation activity. However because of the complexity involved and the amount of change likely to be experienced the mechanisms will require refinement during Active Integration and will need finalising at the start of the maintenance window.

Support

Teams undertaking downstream activities will require support; they will need access to the expertise built up in the Integration Teams. As items become integrated and available to the downstream functions new and unfamiliar territory starts to be explored. They have not seen this functionality before, they do not know how to distinguish different sorts of failure, perhaps even whether it is working or not. Without effective support for these teams the momentum built up by an effective integration can soon be lost.

Support for downstream activities encompass:

- Knowledge transfer. Briefing the teams on what has been delivered, how to work with it and on any known limitations. Providing an explanation of how to spot indicators of problems and how to interpret them to identify the probable source. Knowledge transfer helps to reduce the overall level of demand for ongoing support as it improves the autonomy of the downstream teams.
- Problem filtering. Rapidly assessing whether a reported problem is a red-herring, something arising from incorrect use or process, or is a system problem requiring attention.
- Problem diagnosis. Identifying the nature of each real problem. Is it an integration issue or a problem sitting within a single components? What are the details. It is necessary to determine and capture exactly what is happening and why it is wrong.
- Integration problem workaround provision. For integration problems with large impacts alternative methods of working that avoid or mitigate the problem should be provided.
- Integration problem solution. Again for integration problems with large impacts the team needs to provide leadership of the resolution of the problem.

End Game

The end game for Support and Maintenance is to ensure that not only is the item integrated at a point in time but that beyond that at any time it has to be used in downstream activities its use is smooth and effective. There is no disruption of downstream activities related to the item. This includes disruption because it has stopped working and also disruption because others do not understand what it does, how or functions or what to do with it.

References

1	Integration; the missing link!	SQC Technology Ltd	2010
2	Integration; adopting a new discipline	SQC Technology Ltd	2010
3	Convergence; a key requirement for IT programme success.	SQC Technology Ltd	2010

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.