Managing a Production Environment

Prepared For:	**General Release**
Author:	**Geoff Halprin**
Reference:	**GHOST-PROCESSES-0110**
Version:	**V1.00**
Date Created:	**17 March 1997 17:42**
Date Modified:	**24 January 1998 09:17**

A SysAdmin Group Technical Report

Abstract

Management of a production computing environment is all about the goals of *Reliability*, *Availability* and *Serviceability* (RAS). These have been the benchmarks for evaluating system management practices since the mainframe environments of the 1960s.

In order to attain the goals of RAS, we must seek to maximise *predictability*. The quest for predictability is fought on three fronts; *Standards*, *Processes* and *Technology*.

In this paper I examine the processes that should be used to implement a total quality solution to the management of mission critical production environments.

Table of Contents

1.0 Preface

1.1 Scope

This paper is about the Processes that are required for the predictable, formal management of a production computing environment, specifically the Change Management and Production Acceptance processes.

1.2 Intended Audience

It is assumed that the reader is involved in the management of production computing resources at some level, either at the coal face as a system or network administrator, or at the management level.

This paper is a general discussion paper, and does not use concepts or terminology specific to a particular operating system or environment, except where necessary to illustrate a point.

1.3 References

[1] "Peopleware," Tom Demarco and Timothy Lister, 1987. ISBN 0-932633-05-6.

[2] "Change Management," Michelle Trout. The MOSES Whitepapers.
(Massive Open Systems Environment Standards).
http://www.uniforum.org/news/html/publications/techpubs/moses/start.html

1.4 Acknowledgements

1.5 Change Control

Version	Release Date	Author	Comments
V01.00	24-Jun-97	Geoff Halprin	Initial Release
V01.10	24-Jan-98	Geoff Halprin	Minor updates

2.0 Introduction

Management of a production computing environment is all about the goals of *Reliability, Availability* and *Serviceability* (RAS). These have been the benchmarks for evaluating system management practices since the mainframe environments of the 1960s.

These goals were addressed in the mainframe environment primarily by one manufacturer dictating standards (both good and bad). This problem is far more complex in the open systems world where there are many manufacturers and standards bodies, and further exacerbated by the sheer scale of the distributed environments which we must manage. Such flexibility requires careful management. We must devise a model for the flexible, disciplined management of distributed computing environments.

In order to attain the goals of RAS, we must seek to maximise *predictability*. The quest for predictability is fought on three fronts; *Standards*, *Processes* and *Technology* (SPT).

With **Standards**, we seek to improve predictability through consistency. By improving the conceptual integrity of a site, we reduce downtime due to trouble-shooting and entropy. We keep hosts, subsystems and software versions in sync in order to support a less diverse environment.

With **Processes**, we seek to improve predictability by ensuring that system maintenance activities follow known paths which include quality assurance steps such as peer review, impact analysis and deployment planning. This is predictability through planning.

With **Technology**, we seek to improve the manner in which we manage our environment. From simple tools which automate functions and hence improve consistency of results, through tools which seek to reduce the management effort in real terms through a paradigm shift in the way we perform higher level tasks.

So, where RAS are the measures (we can measure quantities like *Mean Time Between Failures* and *Application Availability*), SPT is the strategy by which we seek to improve those measures. We can allocate a cost to downtime, and hence evaluate the effectiveness of our strategies.

In this paper I examine the processes that should be used to implement a total quality solution to the management of mission critical production environments.

3.0 Setting The Scene - The Cost of Downtime

Why write a paper about a subject as boring as Change Management or Production Acceptance processes? Well, let's just take a few moments to answer that.

Let's assume that you are managing a normal *Prime Shift* computer operation (8am to 6pm, Monday to Friday). That's 10 hours a day, 5 days a week, 52 weeks a year; or 2600 hours per year. (A 24X7 operation is 8736 hours per year, so multiply these figures by 3.36.)

At 95% availability, you are allowed 130 hours of downtime per year, 10.83 hours per month, 2.50 hours per week. At 99% availability, this drops to 26 hours per year, 2.17 hours per month, or just 0.50 hours per week.

Now, let's assume a company employs 100 engineers at an average of $50k per annum. This is a payroll of $5million, which is around $2000 per hour (assuming they work a 10-hour day).

So, on salaries alone, a downtime of 5% means people unable to use the system for 130 hours per year, costing your company $260,000 in lost time. If your availability is increased to 99%, this figure drops to $52,000 - a saving of $208,000. And this is solely based upon salaries.

It turns out that this estimate might be way down the low end of the scale. I have one customer who estimates their cost of downtime to be around $500,000 per day. This would make the above improvement yield an annual saving of $5.2million.

This paper is about predictability. Downtime is a fact of life. Trying to minimise it is only one aspect of reducing this cost. If we can predict when systems will be unavailable, then this will translate into increased productivity (people can plan to do something else), decreased stress ("What do you mean I just lost 8 hours work?"), and a more enjoyable work environment in which you finally have time to improve the quality of life (for yourself and your colleagues and customers).

3.1 Proactive Workflow

Being a system administrator shouldn't be solely about fighting fires - it should be about designing real, systemic solutions to problems before they arise.

The processes presented in this paper are the primary mechanism for shifting the balance of system administration workflow from being reactive to being proactive. By doing this, you gain control over your destiny as a system administrator. You start to find the time to craft those tools you so desperately need, time to find out more about what the user's requirements and expectations are, and time to evaluate new technology, and plan for future upgrades.

So, read this paper with an open mind as to what it can really do for you.

4.0 Managing a Distributed Computing Environment

There are three key processes which form the basis for the quality management of a modern production computing environment;

1. Production Change Management (CM) Process.

   The **Change Management** process is intended to maximise the availability of the *existing* production environment. It forces the careful planning, peer review, post-change testing and user acceptance of any changes to the production environment.

   No matter how "stable" a production environment is in theory, in practice there is always ongoing change taking place which does not affect or augment the underlying functionality of the system. The CM process ensures the integrity of such changes by forcing careful planning and impact analysis of any proposed change. It guides support staff through a controlled learning exercise with respect to the *potential impact of a change on one or more elements* of the production environment.

2. Production Acceptance (PA) Process.

   By contrast to the CM process, the **Production Acceptance** process is intended to guide support staff through a controlled learning exercise with respect to the *introduction of a new element* into the production environment.

   The PA process provides a framework for introducing change into the production environment in a manner which is controlled, predictable and auditable. This process seeks to ensure maximum availability of systems and maximum customer satisfaction with a minimum amount of ongoing intervention by support staff. This involves learning what it means to manage and support the new product, and then introducing it into production in a controlled manner.

3. Problem Management (Helpdesk) Process.[1]

   No matter how we might try, and even with the help of the CM and the PA processes, things will always go awry. In such a case, it is a user who will often notice the problem first.

   There needs to be a clearly defined process for the accepting, handling and tracking of all user complaints, requests and suggestions such that they can be reacted to according to defined criteria, and in a quality assured manner.

   The PM process should not only ensure timely response to user problems, but should provide valuable statistics to management on the progress and effectiveness of support staff, and the satisfaction of the user base.

1. Due to space restrictions, and the already exhaustive coverage of this topic elsewhere, I will not address the PM process in this paper (other than as it provides context to the other processes).

5.0 Process Context

As described in the previous chapter, there are three key processes which contribute to the quality management of a production computing environment. The diagram below illustrates the (simplified) overall process flow and interaction in a normal production environment;

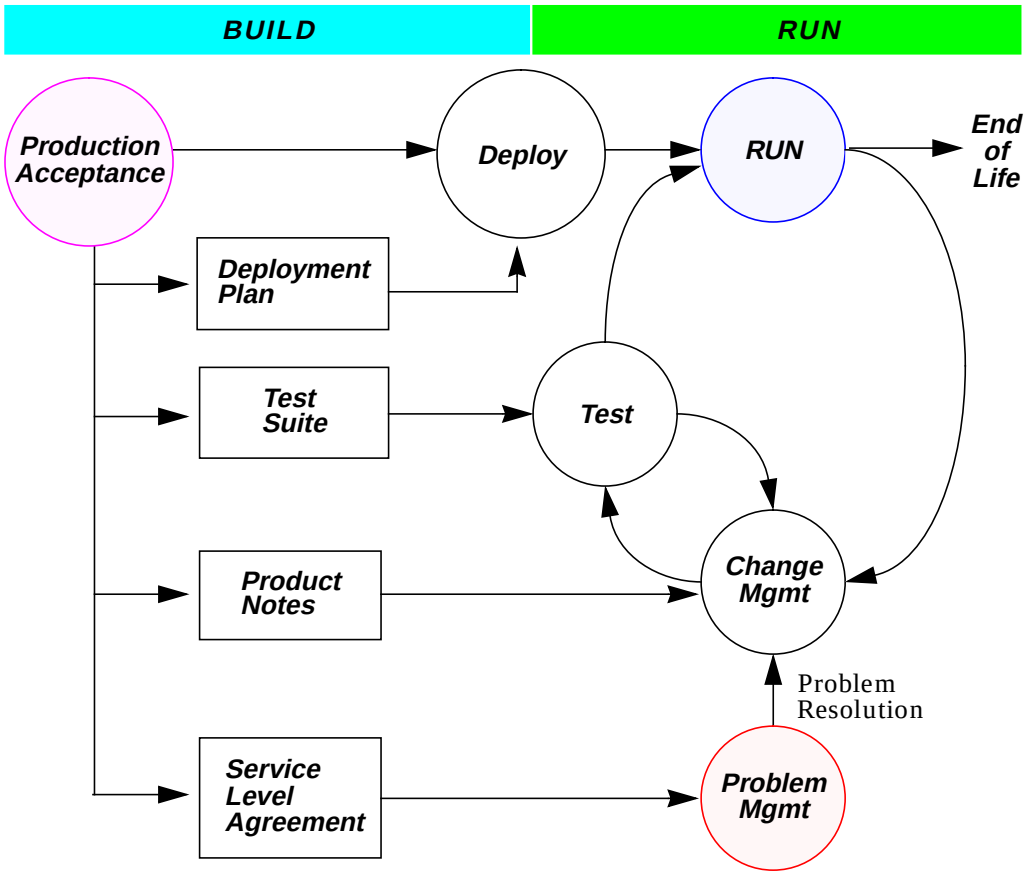

Figure 1 - Process Context

An operational computer system continues to operate in the RUN state until a defect is noticed or a new element is introduced. If the user notices a defect, then the Service Level Agreement is consulted to determine the level of support that should be provided. Assuming that the problem must be fixed, this will then involve changes to the production environment in some form. These changes are controlled by the CM process.

When the Change Management process is invoked, any aspects of the system which are within the affected scope of the change must be tested for correct working behaviour (regression testing) before the change is deemed complete.

When a new element is being introduced into this environment, it is put through the Production Acceptance Process. This guides support staff through the creation of key documentation which controls the entire life-cycle of the product.This documentation includes a deployment plan, a regression test suite, and a product signature, architecture and technical description.

These documents then feed into the other key processes involved in production management.

6.0 The Change Management Process

The first of the processes we will examine is the Change Management process. There is a higher public awareness of the need for a CM process than for the other processes described in this document and these other processes all feed into the CM process, so it makes sense to discuss change management first.

6.1 Why do we Need a Change Management Process?

The principle goal of any change management process is to ensure the stability of the existing production environment by forcing the careful and thorough impact analysis, planning, peer review and testing of any change to the production environment.

Without such a process, production management consists of a never-ending series of ad-hoc, undocumented and unplanned changes, all of which dramatically add to the entropy of the system. It becomes easier to replace these systems than maintain them. This in turn leads to system support staff being entirely reactive, going from one fire to the next.

Indeed, this is usually the most obvious sign of problems at an organisation. System support staff running all over the place, not documenting what they do, not discussing problems with each other, and generally living a fairly stressed existence.

By contrast, using a change management process offers several clear advantages:

- It allows us to solve a problem once. This means that we aren't re-diagnosing problems, and always re-fixing the same thing.
- It allows us to learn from previous experience. By recording and reviewing the process, we can learn from our mistakes and seek to further improve uptime.
- It ensures that the customer is kept informed. This is no small advantage.

Try to answer these questions:

> How did I do it six months ago? Can someone else do it next time?

If you can't, then you're in need of a change management process!

6.2 The Change Management Process

The single most important characteristic of a production operations environment is **predictability**. Most users don't mind downtime if they can plan for it, but these same people will jump through the phone at you if it just happens!

The only way to achieve predictability is through controlling all changes to the production environment, including the steps of;

- Planning the change,
- Testing the change,
- Backing-out an unsuccessful change, and
- User acceptance of the change.

The Change Management process governs the change life-cycle. It enforces appropriate planning to take place before any change can be effected to the production environment and controls the application of that change to the production environment.

6.3 Change Management 101

All changes must begin their life-cycle as an unknown amount of work with unknown effects. Thus, the initial emphasis on change management is to perform a controlled analysis and impact study of the change before any implementation work can be commenced. Resulting from this analysis will be a plan of attack, and this attack must be reviewed by a peer to ensure the quality of the solution. Of paramount importance in all this is to also determine how to test the correctness of the change, and how to back-out the change if it does not succeed. Of course, every change should include user acceptance of the change.

From this we see a very embryonic change management process:

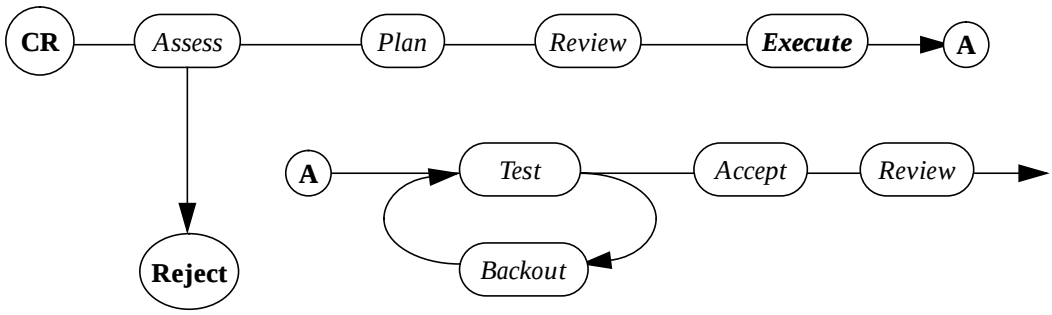

Figure 2 - Embryonic Change Management Process

1. The change begins with a Change Request (CR) being submitted. This is the basic *unit of work* for a change management system. We will examine the CR in detail in the next section.
2. Each new CR must be assessed so as to determine the *scope of change* and the overall *priority* of the change. We will return to these concepts shortly.
3. Once a change has been assessed and resolution authorised, we must plan the change. i.e. How we will effect the change onto a production system; when we will schedule the change to occur; which hosts and applications will be affected by the change; and what actions we must therefore take with respect to those applications.
4. Generally, one person will create such a plan alone. The single most valuable form of quality assurance is to have a peer review that plan and suggest changes or omissions.
5. After these preliminary steps, we are ready to execute the changes as per the plan. Copious notes should be kept for review.
6. Once the changes have been made, we must test (a) that the changes worked correctly, and (b) that all related systems (see *scope of change*) still work correctly. If this is not the case, then we must backout the changes, and re-test that original functionality has been restored.
7. Finally, the user must accept the changes as correct, and we should review the process to see what we have learned.

6.4 Controlled Learning

A major advantage of recording all this information is that once we have learned what is required to make a change, and now have that knowledge on file, we can re-use it when next

we have a similar change. We can further refine and tune it over time, and hence save enormous amounts of personal effort, and improve our rate of success.

6.5 The Change Request

The life of a change begins and is tracked through the Change Request. The CR is a form (paper or electronic) which records the progress and decisions made during the CM process.

6.5.1 Priority, Scope and Severity

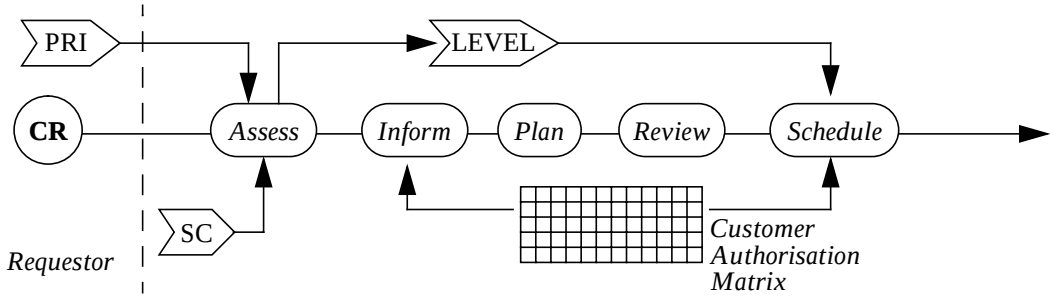

Figure 3 - Assessing a Change Request

Each Change Request has a ***priority***. This is a coarse-grained priority set by the submitter at the time of submission of the CR. This priority reflects the severity of the problem in terms of its impact upon the submitting user. This priority should, hence, be a simple choice based upon criteria such as resolution time. Something like:

P1 Critical. Resolution is required within 4 hours.

P2 High. Resolution is required within 48 hours.

P3 Medium. Resolution is required within 1 week.

P4 Low. All other CRs.

Alternatively, schemes are used which require the submitter to qualify the ***impact*** that the problem is having on them. I do not recommend these more qualitative approaches.

Next, there is the ***scope*** (aka *severity*) of the change.This is a classification of the breadth of the applications, systems, user communities or sites which are affected by this change. i.e. Does this change affect a single user, or the main production host and hence the entire user base? An example might be:

S1 Major site / division affected.Critical functionality lost. No work-around.

S2 Multiple people directly affected. Critical functionality lost. No work-around.

S3 Multiple people affected. Major functionality lost. Staff can continue most work functions.

S4 Failure only affects a single user.

These inputs of priority and scope combine to determine the ***level*** of the impending change. This is discussed in the next section.

6.6 The Change Request Form

All of this information, planning, testing, etc. must be recorded for quality control. Quality Assurance, as specified by ISO-9000 consists of two parts; known processes and records of execution of those processes. The Change Request form is the basic record of the execution of the CM process.

A CR form should record the following information:

1. **Description of Change**. A brief description of what the change is and why it is being performed.
2. **Effect on Availability and Customer Impact Statement**. Details what applications and systems will be made unavailable during the performance of this change, and what impact that will have on the user base.
3. **Risks**. Details what risks are involved in performing this change and what risks there are to the success of the change. A risk analysis should be included.
4. **Prerequisites**. What needs to be in place prior to this CR commencing. This should be a table so that each entry can be signed off as ready. Most importantly amongst the prerequisites is an appropriate form of backup, as this is usually an integral part of a backout plan.
5. **Change Plan**. What actions are to be taken to effect the change? Include as much detail as is necessary to avoid any possible ambiguity when it comes time to follow the plan.
6. **Backout Plan**. What actions are to be taken to backout the change in the event of a test failure or time plan over-run. This should include details of the conditions under which the backout plan must be executed.
7. **Test Plan**. This should include a list of pre-change tests, post-change tests, and post-backout tests. These tests should be derived from the relevant product regression test suites, as detailed in the Production Acceptance process. See Section 7.0 ("The Production Acceptance Process") on page 17 for more details.
8. **Documentation Updates**. This section should list any documentation which requires updating as a result of this change.
9. **Customer Authorisations**. This should be a table listing the required approvals before the change can commence. This table should have room for the appropriate signatures.
10. **Customer Acceptance**. This is a table for the user acceptance of the change as passing post-change acceptance tests. This table should have room for the appropriate signatures.
11. **Change Notes**. A large section for recording any progress notes whilst executing the change. Always record any deviations from any of the above plans.

6.7 The Customer Authorisation Matrix

When managing production services for a large organisation, it is most often the case that different applications and data are used by different user communities, with some common infrastructure, applications and data which is used by all communities.

It becomes vital, then, to build up a list of the managers responsible for each area, what applications they use, and indirectly which systems they therefore rely on. This information is built up into a Customer Authorisation Matrix.

This matrix serves two purposes:

1. When a change is pending which requires some application or host downtime, we can quickly ascertain which user communities will be affected by the change. This in turn tells us which managers to coordinate the downtime amongst, and who must authorise this downtime on behalf of the affected users.
2. When a system breaks, we can immediately identify which user communities are affected, and contact the appropriate managers so that they know we are working on the problem. We, of course, also know who to contact when normal services have been resumed. This pro-active communication maintains a strong communication path and high level of trust with the user communities.

6.8 The Change Control Board

The Change Control Board (CCB) is the controlling body for CR processing. The CCB is responsible for the authorisation and coordination of CRs. Their duties include:

- Authorising CRs. Reviewing the CR, its scope, priority and impact, and assessing this against corporate priorities. Just because someone submitted a CR, doesn't mean that it must be performed! The CCB must decide whether a CR is worthy of the resources required and associated system impact. This authorisation should occur before any 'heavy' work is done on CR preparation.
- Grouping / un-grouping of CRs. It is often useful to group the execution of CRs together, such as those which will occur during a maintenance window. In these cases, the CRs should not work on related systems or products. (This is an attempt to minimise downtime, not combine CRs.)
- Coordination of down-time across platforms and CRs. Where an organisation has multiple platforms which it supports (e.g. Unix, NT, MVS), then the CCB must coordinate changes across platforms to minimise the impact on the user base.

The CCB must approve each CR prior to execution (apart from emergency changes), and ensures that appropriate preparation has been performed, that customers have signed off on the CR and that pre-requisites have been fulfilled.

The CCB should meet weekly to assess new requests, and on-demand for more urgent requests.

6.9 Performing Changes - The Test Lab

There are several very important strategies for improving quality and further improving the likelihood of any given change being successful. As mentioned previously, the single most important of these is peer review.

A second strategy is the trial run of a change in a controlled, non-production environment - the test lab. The lab should be configured to reflect the existing environment, and then the CR put through a trial run.

Note that the amount of gain is proportional to the complexity of the change. For example, it is highly advisable for evaluating new operating system releases or major patches to perform such trials to ensure functionality is not lost.

6.10 Classes of Change - The ESPA System

As was seen in the overall process context diagram, a CM process takes a *Change Request*, performs some work to process that request and (hopefully) returns the system to a fully working state. It turns out that there are several distinct classes of change, and each must be handled differently.

The ESPA CM system defines four classes of change which form a natural hierarchy. These four levels work together to define a comprehensive approach to production change management:

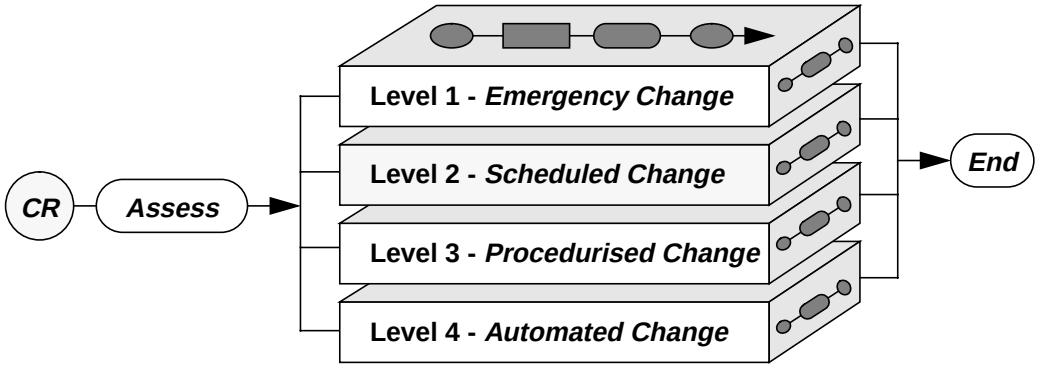

Figure 4 - The Four Levels of Change

Scheduled (The default). The standard level assigned to any change is level 2, the *Scheduled Change*.

This can be characterised as a change which is new (has not previously been performed), and so we must investigate and plan accordingly. The principle role of the Scheduled CM process is that is guides us through a *controlled learning exercise* as to what is required to ensure that the change is successful.

Emergency. This is the process we invoke when something has broken (a major component of the production system is no longer available) and, hence, we do not have the privilege of planning the change. In this case, rather than creating and reviewing a plan prior to execution, our priority must be to fix the problem first.

Instead, we perform the peer review step in the form of a downtime conference, and perform that review after the repair has been performed, in order to ensure that we have indeed resolved all problems and correctly brought the system back into a fully operational state, with known outstanding issues. Every incidence of downtime must be dealt with according to this process.

Procedurised. Those tasks which occur more than once (such as replacing a disk drive, adding a user, migrating Oracle database tables between tablespaces) can and should be turned into procedures.

Once a change has been through the scheduled change process, and we have successfully completed it and noted any failures, then we have completed the necessary controlled learning exercise. We now know how to successfully complete this type of task. Hence, we can take the lessons learned and the plan developed, and turn it into a defined, repeatable procedure.

Automated. Finally, those procedures which, by their nature, we execute many times over can be automated to improve consistency and reduce execution overhead. (You can't automate a procedure you haven't yet defined! At least, not if you want it to work.) Obvious examples here are things like: adding a user, adding a new disk drive, or removing a newsgroup.

6.10.1 Scheduled Changes

The process we developed in Section 6.3 ("Change Management 101") on page 9 was a basic Scheduled CM process.

Scheduled changes can be further broken down into sub-levels based upon such criteria as resolution window. For example, an organisation may deem it appropriate to have five *scheduled* categories: 48 hours, 7 days, 14 days, maintenance window, 60 days. Their Change Management Policy might read:

Level S1 **High Priority**. This level is appropriate for a change which will affect a major production component, but which must occur within a short time frame (48 hours). This might include replacing a disk which is on the way out, or migrating some software between disks to avoid an imminent space problem. This level can be characterised as appropriate when we must provide a limited negotiation with the user base as to when the downtime will occur. ("We have to operate before your appendix explodes!")

Level S2 **Medium Priority**. To be resolved within 7 days.

Level S2 **Low Priority**. To be resolved within 14 days.

Level S4 **Maintenance Window**. To be resolved during the standard monthly maintenance window. This will be the first Wednesday of each month between 18:00 (6pm) and midnight.

Level S5 **Enhancement**. To be resolved within 60 days.

6.10.2 Un-Scheduled Changes

A careful look at the Scheduled CM process reveals that it is only appropriate for those changes we have the privilege of scheduling and planning in advance. Clearly a number of changes do not fall into this category, and are reactive to an incident which has already happened and already compromised system functionality.

A modified change process is required for these situations:

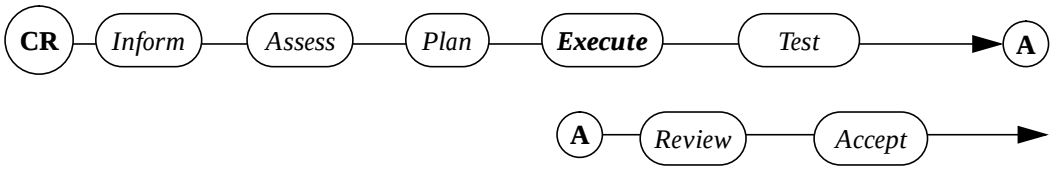

Figure 5 - The Emergency Change Management Process

The primary difference is that rather than peer review and user sign-off happening before the change is effected, we repair whatever is broken (execute), then hold a ***downtime conference*** to assess what happened, whether our reaction was appropriate and sufficient, and what issues remain as a result of this problem. The other addition to the change process is that we must be sure to inform the customer immediately when there is a problem.

When downtime is encountered, the general process should be;

1. Trouble-shoot and repair / work-around the problem. Return the system to an interim working state. Take copious notes on the activities undertaken during the trouble-shooting and repair of the system. (Where feasible, make use of the Unix script(1) command or similar.)
2. Prepare a list of follow-up issues that were uncovered or initiated as a result of the downtime and / or an interim solution.
3. Hold the Post-Mortem (Downtime Conference).

6.10.3 The Downtime Conference

Each incident of server or application downtime must be thoroughly investigated by a composite team of support staff to determine the cause of the downtime, and any courses of action to reduce the likelihood of similar downtime in the future.

The emphasis of these conferences is team learning from mistakes, so that we can continue to improve customer service levels.

This conference should involve a number of support staff. A copy of the incident report (minutes of this meeting) should be circulated to all members of the support team as well as to the management.

Prior to the downtime conference, the primary consultant who worked on the resolution of the problem should prepare a draft of the incident report, as per the agenda below, and previous reports. This will help the conference run more quickly and smoothly, and minimise the impact of these conferences on other members of the support team.

1. Assign incident number.
2. Summary of incident.
3. Current system status.
4. Investigation walk-through. What actions were taken? Are they sufficient? Are there any unplanned side-effects?
5. Incident follow-up. What outstanding issues remain from this downtime.
6. System Resolution. How do we prevent similar downtime in the future?

The Downtime Conference is a powerful tool for identifying trends, issues which require further investigation, and systemic cures to problems.

Warning: Do not attempt to execute other pending CRs whilst a system is down due to a fault. You're sole aim is to follow the Emergency CR process to bring the system back up, then execute other CRs as per the approved schedule. Attempting to group any other CR with an Emergency CR is asking for trouble in the form of prolonged, unplanned downtime.

6.10.4 Procedurised Change - Learning from History

The above change processes both deal with *controlled learning*. We need to ensure the quality of our work by planning the impact of that work. This is an unnecessary repetition, however, where we have already performed such a change, and know the answers.

In such cases, we should be able to document what we did last time, along with what we learned in the reviews, and turn this into a procedure to follow whenever a significantly similar situation arises.

Such procedures must still include all necessary customer coordination and inter-CR downtime scheduling, but the actual planning and review steps become much more streamlined, and the peer review is eliminated greatly optimising the overall process. Procedurising a change does not affect the level of authority required to perform the change either. This can be documented and enforced by policy as appropriate to the organisation.

Of course, we still need to review the procedure to identify when a process is no longer appropriate, or must be enhanced due to a changing environment.

6.10.5 Automated Change

Finally, where a particular change request is frequently executed; then once we have procedurised the change and tuned that procedure over a few executions, it becomes possible to reduce the overhead of executing that procedure by automating it. It is a waste of time to attempt to automate a procedure which has not been defined, and is therefore not properly understood.

7.0 The Production Acceptance Process

The PA process is the natural counterpart to the CM process. Where the CM process addresses the problem of maintaining existing functionality, the PA process addresses the introduction of new functionality.

The PA process provides a framework for introducing change into the production environment in a manner which is controlled, predictable and auditable. This process seeks to ensure maximum availability of systems and maximum customer satisfaction with a minimum of ongoing intervention by systems support staff. The PA process contributes to this by addressing issues such as the following:

- How are new products introduced to a stable production environment?
- How are applications successfully transitioned from development to production?
- What are the training, security, maintenance, availability, output, and other requirements of new products in a production environment?
- Who is responsible for support, security, hardware and software maintenance, etc.?
- What host resources are affected by the installation of the new product?
- How does one recognize when the product is not working as expected?
- How does one add users to this product?
- How is backup and restore of this product accomplished?
- What are the disaster recovery requirements of the product?
- How does this product interact with other products?

In short, the PA process guides support staff through a controlled learning process to ensure that they understand the effect of a new product on the production environment before they introduce it into that environment. In short, it answers the question;

> "What does it mean to say 'yes' to this product?"

Before we commit to supporting something, we need to know what that means to the entire support organisation. The PA process ensures that we understand what it is that we are about to commit to.

The alternative to the PA process is to just introduce a new product, and discover its idiosyncrasies over time as errors arise. Usually, these findings will not be documented in any coherent manner leading to greater repetition of effort, decreased consistency, and increased entropy.

It may seem like extra work to perform the PA process, but the fact is that you will always incur this expense; the PA process merely captures it all up-front and records the results where they can be found rather than this being a hidden cost. Using the PA process, you will also exert far less effort for a greater return.

7.1 The Process

It would appear that the PA process will make production management predictable, simple, controlled and perhaps [gasp!] a little less stressful. So, what is it?

The PA process consists of a small number of well-defined steps. Each step takes certain inputs in the form of guidelines and document templates, and generates one or more records. These records either form an input to another step, or are filed as part of our new product knowledge.

Phase	Step		Outputs
Proposal	1	Create Product Proposal and Quotation	Product Proposal
	2	Plan Project	Coarse Project Plan
GATE 0			
Negotiation	3	Create SLA	Service Level Agreement
	4	Create Acceptance Test Plan	Acceptance Test Plan
	5	Create Deployment Plan	Deployment Plan
	6	Review Quote	Revised Project Plan
GATE 1			
Investigation	7	Analyse Product	Product Notes
	8	Stress Test	
Revision	9	Review Deployment Plan, Test Plan and SLA	Revised Plans
GATE 2			
Deployment	10	Product Installation	Installation Record
Operation	11	Initial Operation and Tuning	Operational Observations
Acceptance	12	Customer Acceptance	Customer Sign-off
Review	13	Review Process	Process Review Minutes.

Figure 6 - Overview of the PA Process

The intention is that a single binder (or the on-line equivalent) is built up per product, and this binder is then continually referred to and updated as the product is supported over time.

When there is a problem with a product we can immediately run the acceptance tests detailed in the binder to determine whether the system is functioning as per customer specifications. Where it passes these tests, but still “fails” customer requirements, then we must review the test plan, and perhaps also the SLA.

Moreover, when we are performing any change to production (via the CM process), then we can identify all products which are in scope (affected by the CR), and hence we immediately have a list of known tests to perform with respect to those products. Further, we will always run the same tests with respect to a products functionality. This is a huge win in terms of quality assurance.

By creating detailed product descriptions, we can quickly understand the basic nature of the product and how to trouble-shoot problems with it.

A second important aspect of the PA process is that each new product is passed through this process, and so each time the PA is executed we have created a self-contained project. This project mentality is useful; in seeing the product successfully through the PA into production we have completed the project. This gives you, management and the customer a sense of achievement.

The PA is not a Heavy Weight Methodology

At this point it should be made clear that the PA process described is a guideline. It is a little 'm' methodology, not a big 'M" Methodology [1]. You should always apply it with intelligence and discretion.

Clearly, some products require vastly more effort to analyse and control than others. It would not be appropriate to spend the same effort bringing Samba into production as Oracle. Similarly, a simple product being installed on a single machine will not require the same complexity of project plan and deployment plan as a new piece of accounting software to be deployed across a number of sites as a replacement for an existing system.

The intention of the PA is to guide you through a process, so that you can make those decisions, and be prompted to look at critical issues in making these determinations. As with everything else, there is no substitute for experience. The PA process will be more wieldy to use at first as you are unfamiliar with it, but with time will become second nature.

7.2 The Steps of the PA Process

The above overview does not reveal the amount of work involved in each step, nor the real nature of that work. We must look at each step in turn in order to appreciate what it is trying to achieve.

1. **Create Product Proposal and Quotation**

   Life for a new product must begin with a user requirement. At that point a product proposal is written detailing the business case for the new product. Also documented is the priority of the product, and the time frame within which it must enter production.

   This step formally recognises the requirement for a new product, and hence sets management expectations with respect to the work involved in executing the PA process for this new product. This avoids the trap of management just expecting a new product to be introduced "in passing."

2. **Plan the Project**

   It is very important that we control the effort required to bring a new product into production. As was stated in the introduction to the PA, the cost of deploying a new product, and the cost of the ongoing support of that product are non-trivial, and these generally go un-identified and un-controlled. This leads to support staff overload.

   If more than a week of time is going to be required, then it is definitely worth a proper project plan. Depending upon the nature of your organisation, you may wish to charge the customer for time spent on this project. In such cases, a customer will more often than not wish to see a project plan and budget.

3. **Create a Service Level Agreement (SLA)**

   The most overlooked part of a non-PA regime is that of setting customer expectations. Do they expect the product to be available 24 hours a day? Of course they do - if you haven't pointed out how much that will cost.

   The SLA is the basic document for agreement between the customer (external or internal) and support staff. This document is created in close consultation with the customer because both the customer and the support staff must be happy that they can meet the levels of support promised in the SLA.

   The SLA becomes the benchmark by which quality and quantity of future service will be measured.

4. **Create an Acceptance Test Plan**

   One of the most powerful concepts behind the PA is that right up front we define some tests, in agreement with the customer, which constitute the benchmark for whether the product is behaving as expected. Whenever there is a suspected problem with that product, we can immediately identify whether this baseline of functionality is being met.

   This gives support staff a far more open and powerful way for dealing with customers. If you can clearly show that the system has met their defined requirements, then they will be more willing to help you tune those tests and the system behaviour to match their changing requirements. A more congenial and respectful relationship will generally result.

   These tests are far more than just acceptance tests performed once as the product is placed into production. Whenever a CR is processed which might affect the functionality of this product, we re-run these tests to ensure that it is behaving as expected. We generally refer to such a set of tests as a *regression test suite*.

   As more products are run through the PA process, we build up a more comprehensive set of regression tests such that we can easily test system status, and SLA compliance.

5. **Create a Deployment Plan**

   Bringing a new product up under isolated conditions is a relatively straight forward exercise. You can probably even just follow the instructions that came with the product. But introducing a product into an existing production environment is far more difficult.

   Even armed with full knowledge of the product (as discovered in step 7 below), there are many changes that will effect functionality of other products across a number of target hosts.

   Deployment of the new product must, therefore, be carefully planned. Events such as downtime must be coordinated. Where a product is to be deployed across multiple hosts, we must do a partial deployment and test for unexpected side-effects before continuing with a wider deployment.

6. **Review the Quotation**

   At this point we have a far better understanding of the requirements of the product, and so can review the quoted effort and costs of accepting this product into production.

7. Analyse the Product

This is the heart of the PA process. Here we perform a sequence of steps to gather knowledge about the product and its interaction with other products.

Using a sociability laboratory (see below), we create a virgin system image, and snapshot that system (Tripwire is an excellent tool for this). We then load the software onto this test platform, and re-snapshot the system to detect changes. This tells us what parts of a host the product touches during installation.

We then run the software and again snapshot the system. This tells us about what the product touches when running. All of this is on a test platform, isolated from production hosts.

We then need to scour the product manuals and ascertain the overall architecture, component interaction and data flows of the product; what processes does it comprise? How does data move between these? What are the basic data flows of the product into various data areas? What accounts/groups does it require? How is the application started up cleanly? How is it shutdown cleanly? How do we determine its current status? What regular housekeeping is required?

8. Stress Testing

A vital step in the acceptance of applications with a wide user base is the true stress testing of the product to ensure it meets the needs of the intended user base under load. It is quite astounding the number of problems which first surface under true production conditions, and stress testing is the primary tool in attempting to reduce this occurrence, and the ensuing nightmare of eleventh hour changes.

At the completion of the product analysis step, we have a fully installed product in the SocLab, on hardware similar to the final production hardware. (See Section 7.4 ("The Sociability Laboratory") on page 22 for more on this.) We can now proceed to stress test the product to determine how it behaves under a representative load.

Of course, it is usually impossible to test a product using an actual load, so instead we often need to construct test scaffolding which simulates the user load in some appropriate manner. Discussion of benchmarking tests is beyond the scope of this paper.

9. Review SLA and Deployment and Test Plans

Having now created a knowledge base about this product, we are in a very good position to proceed with the installation. Before we do so, however, we must review the SLA, deployment plan and test plan to determine if they are still appropriate, given this new knowledge.

10. Install Product

Now, with that major work behind us we can finally deploy the product according to the deployment plan, with a reasonable level of confidence in the results.

As part of this we must re-test all other products on any affected hosts to ensure that existing functionality has not been compromised.

Note that in a distributed environment, this installation step is non-trivial, and may be quite complex. In fact, it is combined and overlapping with the stress testing and user acceptance testing steps. We will often perform an install in several distinct phases, onto a number of hosts, clients and users as is appropriate, testing under an increasing user load as we go.

11. **Initial Operation and Tuning**

    Once the product has been fully deployed, there will usually be a 'settling-in' period where the product is closely monitored and tuned under a real workload.

12. **Customer Acceptance**

    Finally, we can show the customer the results of the acceptance tests and have them sign-off that the product is in production and working correctly. We now have the baseline for a working production product.

13. **Review Process and Project Closure**

    An important part of quality assurance it the review of the project and of the PA process, and the continuing tuning of that process to better meet the environment.

7.3 The Authorisation Gates

The table that was presented at the beginning of this section (Figure 6 on page 18) had various steps separated by *authorisation gates*. It is often the case in larger organisations that a project does not get complete approval at the start and then just proceeds until done. Rather, there are a number of checkpoints to ensure that the project is proceeding as expected, and that costs are within budget, or that an appropriate budget extension has been authorised.

The authorisation gates are that basic mechanism for review, and hence occur after each revisit of the project plan. Again, this emphasises the need for formal project management on larger mission critical product installations.

7.4 The Sociability Laboratory

In analysing the product, the notion of the Sociability Laboratory (test lab) was introduced. This is an extremely important part of both the CM and PA processes. Wherever we are doing something for the first time, we should do so first in a controlled environment, isolated from the production environment, where any unexpected behaviour will not affect production systems.

For the SocLab to be of most benefit, it should be identical to, or as close as reasonable to, the existing production platform. This will further reduce the likelihood of unexpected behaviour first showing up on the production platform. This is especially true where the SocLab is used during the CM process to test changes before applying them to production, (such as testing operating system or product patches), and for stress testing of applications prior to production deployment.

7.5 Roles in the PA Process

The PA process is best performed by the staff that will be supporting the product in production. It is also best performed as a team of two or more people. This ensures that key knowledge is more likely to be recorded, and we further reduce any key-person reliance.

This is an excellent opportunity for support staff to perform non-reactive tasks, and to learn and practice the planning aspects of production management.

8.0 A Final Word

In this paper I have introduced two key processes. The Change Management process forces support staff to be proactive and plan changes to a system, rather than just execute them and wait for the consequences to become apparent. The Production Acceptance process forces support staff to learn about a product before they support it.

Together these processes provide a powerful tool for the management of mission critical computing environments.

This paper covers a lot of material in varying levels of depth. In order to gain the most out of this topic, I have identified the key concepts which, if you take away and use, will provide the most benefit.

1. **The Production Acceptance Process**. The PA process is the natural counterpart to the CM process. By forcing each new 'product' to be put through the paces, we obtain a far greater level of *quality of service* and this can be measured in terms of application availability. Remember, you will always spend this effort in some form, the PA process just maximises the return on that effort.
2. **Regression Test Suites**. Providing a single consistent set of tests for each product, and then using these same tests each time a CR is processed which may affect that product, will significantly improve the quality of changes.
3. **Downtime Conferences**. Diligently holding downtime conferences after every occurrence of application (not just host) downtime, and seeking to perform a root cause analysis will yield significant improvements to application availability (as long as you follow it up). If you compare the cost of downtime (what do you mean you haven't calculated it?) to the cost of this analysis, you will be quite surprised by the gains.
4. **The Sociability Laboratory** (aka Test Lab). Having a test laboratory in which you can test operating system upgrades, patches, and new products will significantly increase the success rate of changes and additions, and reduce the incidence of downtime.