

Applied application and service provisioning

AUUG 2005 - The conference for Unix, Linux and Open Source Professionals

Jorgen Skogstad, Iain Jardin

Sun Microsystems Australia, 476 St. Kilda Road, Melbourne, 3004 VIC, Australia

{jorgen.skogstad, iain.jardin}@sun.com

Abstract

This paper addresses facets pertaining to ‘*software life cycle management*’ and how application of provisioning techniques can rationalize the management of it. It describes some of the problems and challenges many organizations face today; most of them related to either *people*, *process* and/or *technology*. It also provides some insight into how they may be addressed through the application of provisioning techniques, though not in technical terms. This would be impossible to do within the scope of this paper. The paper also provides some real-life examples of the tangible and intangible benefits an organization could typically expect when moving to apply concepts of application and service provisioning into their environments.

1 Introduction

As more demands are being put on IT to deliver more, faster and to a reduced cost, we need to find ways of delivering short- to long-term improvements. Applying application and service provisioning techniques is an efficient method of doing this. Applying methods and techniques in this space not only solves technical problems, but forces an organization to align towards a common goal of simplifying their IT operation. When addressed properly, all parties within an organization should see the immediate benefits and support the provisioning adoption initiatives. In this paper we delve into many important facets concerning applied application and service provisioning and how they solve many of the problem areas in today’s challenging IT environments. It should be noted though that we are only able to ‘*scratch the surface*’ of these topics as the area itself can be huge. However, we hope to induce elements for thought that people can leverage and relate to their specific day-2-day IT operational environment.

2 Problems in today’s environments

Most organizations dealing with information technology today face a common set of challenges. Most of them can be attributed to either one of these areas; *people*, *process* and/or *technology*. In theory there should be a sound balance of these three elements in any organisa-

tion. This would be a sign of a healthy, mature and organized environment and organization. The reality however is not as bright; we all know that most environments today have grown increasingly complex and hard to manage. This is especially true for organizations dealing with ‘*software life cycle management*’ and many of these organization are (1) developing applications and services inhouse, (2) governing their infrastructure and (3) having an inhouse operations department managing the environment.

The inherent problem in multi-faceted organizations as mentioned above is the focus of the internal departments and the way they work. Take the example of a development organization producing ‘*application and services products*’ that is to be integrated into an infrastructure managed by the operations department. The development department derives business requirements and iterates through a development cycle until a product is complete and ready to be integrated into an environment. The product that has been developed, then typically is handed over to the operations team for integration and continued management. Seemingly straight forward and simple, right?! In theory yes, but what actually happens here in many cases? Within the development department most product development is organized on a project by project basis and have ‘*autonomous control*’ of what happens until it reaches a mature stage where it is ready to be transitioned into an operational environment. This in itself impose a challenge; if

there are no imposed technical requirements for the projects to develop in accordance with, the end result is that whatever they produce will not be *'standards based'* and *'simple to manage'*. Hence; the operations department become *'hostage'* for all the decisions taken by the development projects themselves. In all simplicity, this is an organizational alignment issue. One might say *"Ok. This should be easy to fix. Just make sure the development and infrastructure representatives sit down and agree what happens."* Only if it was that simple; in most organization there are budget constraints forcing short-cuts to be made at every corner. Hence there is no real incentive for this cross department communication to occur.

Out of experience, many will not accept that this scenario describe the environment they are working in. That might be true, as it is a simplification of a typical challenge many organizations face. There are no *'one size fits all'* when it comes to these matters. Associated with the described problem statement above, is the notion that many development methods embeds the provisioning aspects and methods. This is in most cases not true and again ties into which problem spaces the respective areas tries to address; *agile development* vs. *efficient operations*. Almost per definition, development methods and their associated processes do not simplify or *'solve'* the areas of *'infrastructure management'* and *'aligned IT operational processes'*. To drive alignment between the two, requires that *'applications and services products'* from the development department (or project) of an organization should be subjected to a defined, agreed and aligned *release-* and *change management* process. This approach would ensure an efficient IT operation. If this is not the case, there is a great chance that there are gaps and room for improvement.

There is one answer that would solve "all" such organizational alignment issues: *process orientation of the IT organization*. Adopting a pragmatic and process oriented way of working with IT Service Management will, if successfully adopted and implemented, solve these issues. Though it should be underlined that doing this is not simple. It will require substantial efforts from all parties of the organization and it will take time.

So why are all of these problems and challenges becoming increasingly burdensome for organizations? There are many reasons for this, but the most evident reason is, again, the lack of processes pertaining to the management and operations of the architecture. Hence the growing interest within the industry in operational and process frameworks defined by best practice in areas like IT Service Management ("ITSM"), IT Infra-

structure Library ("ITIL") and Control Objectives for Information and related Technology ("Cobit").

3 Addressing the problems

As noted earlier, there is a growing industry awareness around process orientation and how it is required to drive improvement into an IT organization. The adoption of operational processes also enable that certain tasks and functions can be effectively automated. Hence underpin the organizational requirements imposed from the business; deliver more for less.

But in practical terms, what does this mean? Considering the ITIL process framework, it means that the defined *release-* and *change management* processes are underpinned and strengthened by a centralized deployment and change *'engine'* for all areas of *'software life cycle management'*. It is here things start to get complex as this entails organizational *alignment*. This would mean that everyone dealing with software and services *development, testing, deployment, operations* and *management* would have to manage and operate all related tasks through this *centralized core deployment system*. An environment supporting such an organization has certain characteristics, which Table 1 describes.

Table 1 describes characteristics of a well aligned shared services infrastructure model. It describes a typical four-stage release cycle from development to production and what should (ideally) be characteristics for each stage. Elemental to it all is the centralized management of all software and services that goes into the infrastructure; it is all be automated with no manual interaction and/or intervention. Furthermore it also restricts who has access to the different environments within the infrastructure. Effectively the only ones able to *'log on'* to the systems are either (1) testers or (2) administrators; no developers or *'other entities'*. Why? Because one in the ideal situation wants to ensure that the centralized deployment systems is always up to date with all changes within the infrastructure.

Ok, so all of this may not be practically possible today and maybe never will be. However, it is important to bring along as many characteristics of such a model into the implemented model and strategy. It will ensure organizational alignment; both technically and organisationally. This will benefit the organization in the long-term.

So; if one have successfully been able to adopt and implement a provisioning strategy there should be benefits to be reaped? Indeed, so let's investigate some of them.

| Description | Development | Test | Pre-production | Production |
|--------------------------------------------------|-------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------|
| Goals of environment | Functional tests | Functional tests | <ul style="list-style-type: none"> • Functional tests • Production tests • Performance tests | Production |
| Performance testing | Limited | Occasional | When required | N/A |
| Dynamic vs static | Highly dynamic | Dynamic | Mostly static | Static |
| Number of environments | Hundreds | Tens | Few | One |
| Consumers of the environment / service | <ul style="list-style-type: none"> • Developers • Testers • Administrators | <ul style="list-style-type: none"> • Testers • Administrators | <ul style="list-style-type: none"> • Testers • Administrators | <ul style="list-style-type: none"> • Administrators |
| Supporting infrastructure | Cost effective commodity systems | Cost effective commodity systems, but with adequate storage capacity | Systems with production system topology, but possibly scaled down | Production server topology fulfilling production availability and performance requirements |
| Time to deploy new environment (whole stack) | Short | Short | Short / medium | Medium / long |
| Deployment of applications | <ul style="list-style-type: none"> • Automated • No manual deployment | <ul style="list-style-type: none"> • Automated • No manual deployment | <ul style="list-style-type: none"> • Automated • No manual deployment | <ul style="list-style-type: none"> • Automated • No manual deployment |
| Typical availability of the deployed environment | 2-5 days | 2-15 days | <ul style="list-style-type: none"> • 5-30 days • Semi-permanent | N/A (always available) |

Table 1: Characteristics for an environment supporting a staged deployment process

4 Benefits

Knowing what kind of benefits one could expect is important when investigating provisioning strategies. They often define and set the bar as to what one does and when. Assessing and defining what benefits one ideally would like to achieve makes it easier to choose an implementation and project strategy. So what can be expected? Simplistically there are two areas of benefits that can be expected; the *tangible* and the *intangible*. Let's move on and see what they can be.

4.1 Tangible benefits

What are tangible benefits? In short, they are the *measurable, immediate* and *direct effects* you would derive from your initial deployment and project. They are a *key success factors*, since you will have to prove that the efforts put into adopting a provisioning strategy has been worth it. So, what could you typically expect? Well, the simplest and most effective result is the reduced deployment time for your applications and services architecture. The table beneath describes an extract from a real-life implementation where one goal (out of many) was to rationalize the deployment of the organizations standard software packages. Bear in mind that the example

beneath represents only a small portion of the effects from the project. However it descriptive of the typical results experienced from such an implementation.

| Application package | Pre-automation | Post-automation | % reduction |
|-------------------------|----------------|-----------------|-------------|
| Sybase ASE server v12.5 | 90 mins. | 14 mins. | 84% |
| Sun JES Webserver | 120 mins. | 3 mins. | 975% |
| Sybase JConnect | 10 mins. | < 1 min. | 90% |
| IBM DB2 client | 15 mins. | 1 min. | 93% |
| IBM MQ client | 10 mins. | 1 min. | 90% |

Table 2: Examples of reduced deployment time through the use of automation techniques.

There are a few other tangible benefits that are interesting when implementing automation like this; especially around the *accuracy* and *standardization*. Since you have done most (if not all) your legwork in the application *planning* and *factoring* phases, you are ensured that all your standard deployments are 100% accurate. For simple applications and services this might not be *'that'* important, as you might not spend too much time deploying them any ways. However, the picture is different for manual deployments of applications like Oracle, SAP and Sybase. You might be tied down for a day or two getting a system and the service up and running exactly the way you would like it. With automation you will be able to do the same amount of work in minutes, rather than hours and days. To provide a practical example; consider the following deployment work that was done for a financial clearing house recently:

- The Oracle 10g CRS deployed in ~7 minutes.

- The associated Oracle RDBMS deployed in ~10 minutes.

This means effectively that you have deployed a fully available Oracle 10g environment in about 17 minutes. “So, how is this possible?” In short terms; *standardization throughout all layers of the service architecture*. To exemplify:

- Core infrastructure standards; including elements like naming services, ip addresses, access policies..
- Standardized hardware platforms: typically a maximum of three variants.
- Standardized operating systems platforms: typically a maximum of three variants.

Driving standardization throughout all of these layers, will make it simpler and more efficient when trying to automate. Effectively one have to create, in simple terms, a localized ‘*patterns*’ approach for all your IT elements and components. Onto these ‘**patterns**’ you build the automation logic that enables rapid deployment with all it’s added benefits.

“So how can standardization and rapid deployment also drive cost efficiencies into an organization and data centre?” Well; since automation lays the foundation for true ‘*moveable*’ applications and services, one can now increase or decrease the ‘service’ capacity upon need. Hence you can use your infrastructure more efficiently. To exemplify, have a look at the following two cases:

1. Massive ‘short-term’ registration and information services. In certain environments there is a massive peak in registrations and requests for information that requires scalable resources and services. Typical environments like this could be university enrollments, conferences, emergencies and so forth. If one have laid the foundation to automate the deployment of new services ‘on the fly’, one can timely react upon consumer needs.
2. ‘Grid style applications and services’. This can be an extension of the above, but could also be specific services that are deployed into a ‘resource pool environment’ upon need. The basis to being able to do this in real life lies on the ability to componentise and standardize the environment. If all components are known and standardized, it would be a fair task to virtualise and automate them to provide true ‘grid’ style services. Many organizations opt to do elements of this to support their ‘*software life cycle management*’ solution. Effectively having a resource ‘pool’ available to enable a dynamic development and test environment, like the one referenced in Table 1 earlier in this paper.

An area of deployment and provisioning we have not delved into are the areas of bare metal provisioning. I.e. the base platform levels, including the hardware and operating systems. It should be noted that the standardization and automation of these layers share the same tangible and intangible benefits; only with a few differences.

Interestingly the industry trend is to further extend these provisioning areas into complete infrastructure provisioning and virtual is at ion. The major obstacle thus far have been the sheer scope and complexity of existing infrastructures. With the advent of vendor and industry infrastructure patterns, this is about to change, but it will require time for general industry acceptance and adoption.

This is seemingly good (or at least interesting), right? Indeed! However, things get even more interesting in an infrastructure where you have hundreds (if not thousands) of these components iteratively deployed, updated and managed on an ongoing basis. Tie these results into the scenario where you support a highly dynamic environment which is described in Table 1, and the case gets even more interesting. Adding then the fact that anyone can manage deployments and that your typical operators need not be available when performing them. This is true efficiencies brought to the business, but also to the IT resources as they can be leveraged to deliver further optimization and rationalization within IT. The thing to understand is; *a cycle of continuous service improvement is necessary and mandated to have this work in the long-term*. Thus it is important to have ‘*management*’ and ‘*financial representatives/parties*’ understand that this is not just an area for reducing IT costs, but rather an area to optimize how IT delivers value to the business. Sounds like ‘*fluff*’, but it is true! Adopting strategies, tools, technologies and processes in this space requires tighter alignment between all parties in an organization. The good thing is that it brings immediate and tangible benefits to all parties, if adopted and implemented correctly.

It is evident that there are some interesting immediate and tangible benefits from adopting a provisioning strategy. However, there are some interesting other facets that captures the interest of the business and management that should not be forgotten. Why? Well, they will most often mandate and fund the project(s), so they would require persuasive information. When one effectively capture and drives standards through automation, it simplifies the IT environment. This is key to managing long-term cost reduction initiatives, but it will also inevitably aid areas like:

- Disaster recovery

- Adoption and use of cost effective commodity based systems
- Making the organization ‘more’ independent of (primarily) hardware and software vendors.

Why is this? Well, let’s describe each one of these topics.

When all facets of an application’s deployment steps are captured and automated, one should be able to re-deploy it elsewhere without problems. This is a key requirement when choosing to automate an application and/or service. It should be ‘*platform- and system independent*’. One can easily see that this would lay the foundation for a more agile environment where one can re-organize your IT ‘*items/components*’ upon need. This is obviously cost effective. Since one now can manage to redeploy (potentially) any application and/or service at any time to any place, this is an effective way to manage disaster recovery. Upon failure of a system or even a data centre, you can use the centralized deployment system to re-deploy the application and/or service to a standby architecture.

Standardization and automation will enable the use of cost effective architectures based on or augmented by commodity based systems. When assessing the deployment characteristics of applications and/or services, you develop organizational standards for handling them. Hence you can decide on characteristics that enable the use of commodity based systems. A practical example of this are many application server architectures (like IBM WebSphere, BEA and even similar Open Source ones). They can often be implemented in a vertical- and horizontally scalable manner. If horizontal scaling is preferred, you may elect to simplify the deployment configuration of the ‘*nodes*’ in the application server ‘*farms*’. I.e. making them autonomous. This coupled with an intelligent networking architecture where load distributing facilities are available, one has a foundation for a cost effective architecture where commodity based systems constitute the core.

Driving standardization drives independence. There are too many facets to delve into details on this, but a practical example is again the use of application server architectures and the attributed development standards. If an organization elects to use standards as j2ee (or similar), you may have the option of moving your developed applications and/or services between different vendors’ implementations of application servers. This would underpin vendor neutrality and help the organization to keep the vendor(s) at ‘bay’. This is again a cost effective approach which is an immediate benefit. [In many cases this is enabled by using shared services; i.e. decoupling

services and making them reusable as in Service Oriented Architectures (“SOA”)]

4.2 Intangible benefits

Complementing the tangible benefits experienced by adopting deployment automation are all of the attributed intangible benefits and effects. The table beneath describes some of the most typical benefits one could expect.

- **Reduction of fault incidents**
Since deployment automation relies on standardization, there will be less incidents occurring. Typically related to operator errors and mis-configurations.
- **Simplified deployment**
Standards will make automation possible. Standards coupled with automation provides simplified deployments.
- **Faster deployment**
Since deployment actions are captured and automated, they can be replayed and executed faster.
- **Multi-tier deployment**
Ability to execute n-tier deployments. This is interesting as one can model a service ‘*chain*’ and deploy it upon need. When considering an organization that shares characteristics as in Table 1; multi-tier deployment would be beneficial as it could *speed up development and create a more dynamic development-, change- and release environment/process*. As an example; if a developer requires a whole ‘*service chain*’ to perform specific functional tests, then multi-tier deployment may enable a deployment of it within hours. This is often never an option as it would simply require too much from the organization to execute accordingly.
- **Traceability and audit ability**
Adoption of a centralized ‘*software life cycle management system*’ with an associated operations process ensures that all actions are logged (and/or audited). Hence all actions, events and results are traceable and audit able. This is becoming ever more important as legal and regulatory requirements mandates this [see separate section on this later].
- **Change authorization**
Ensuring that the centralized deployment system is implemented and underpin adopted release- and change management processes, ensures that all changes in the IT infrastructure are authorized. (Typically this is augmented with an appropriate identity management system and any other similar and required measure)
- **Platform consistency**
Automation of deployment through provisioning

delivers an immediate benefit of all platforms being consistent. This would benefit and simplify typical operational processes like incident and problem management. Why? Since the organizations resources dealing with incidents and problems would be dealing with organizational standards for the IT environment.

- **Infrastructure standardization**

A key element in an agile environment using deployment and provisioning techniques is standards. Hence most (if not all) elements of the infrastructure would (ideally) be standards based.

- **Version control (and history)**

Using a centralized deployment system would ensure that all versions of software and services are stored and managed. This underpins release- and change management processes, since key to this is to revert to old ‘versions’ if something fails. Essentially version control like this, solves issues related to back-out and rollbacks that are key to managing changes in an environment.

- **Defined and agreed boundaries**

Since the adoption of a centralized deployment system requires the acceptance of all parties involved in the ‘*software life cycle management process*’, their relationships, roles and responsibilities must be mapped out. Hence this drives organizational alignment.

- **Simplified administration**

Administration relating to all areas of ‘*software life cycle management*’ is simplified. Organizational standards, defined processes, automation, policies etc. all contribute to this.

- **Centralized repository**

The centralized deployment system would hold all versions of applications, code, configurations and so forth. In ITIL terms it would be the DSL. [Though the ITIL definition of the DSL does not cover areas related to provisioning systems. The ITIL DSL describes a software ‘package’ repository; like managing the physical copies of software that an organization have purchased. In a provision able environment, we would be dealing with an ‘extended DSL’.]

All of these intangible benefits are difficult to measure in dollars, but should reflect a greater benefit to the organization in the long-term.

5 “So, what is required?”

“So, what is required to in real life do this?” one might ask. Not a simple question to answer as ‘*no size fits all*’. There are a few indications wether automation through provisioning may be a path worth investigating:

- The organization in question is fairly ‘*mature*’; for instance investigating process orientation based on frameworks like ITIL.
- The organization in question is of ‘*fair*’ size.
- The environment has a ‘*sizeable*’ number of services.
- The environment is dynamic (or at least it should be); for example, the organization does a lot of inhouse application and service development.
- There are indications that the environment experiences service downtime and/or degradation due to mis-configuration, manual errors, missing processes and similar things.
- There is an ever important over arching requirement for IT to deliver more for less.

If an environment does not share characteristics with the ones described above (or similar ones along the same lines), applying application and service provisioning into the environment may not be the desired path to walk.

However; if it has been decided that it is a path worth walking; what needs to be part of the equation to manage the project(s)? The foremost question would be the complexity of the environment. Generally speaking, the more complex the automation logic required to be built, the more effort it would be required to develop and maintain it. It should be assessed wether the organization would benefit from implementing it. There are attributed costs for the implementation of such project(s); again the technology and process areas are ‘*simple*’ and can be fairly estimated. The people aspect is another thing. Key questions to raise are: Is the organization mature ‘*enough*’ to accommodate such an implementation? Are the right resources available? Can we maintain it post-implementation? And so forth.

As an esteemed colleague once stated: “Clients should understand and indicate how much ‘*gravitational pull*’ their automation must have to be ‘*deemed successful*’ (more automation gravity = more automation mass = escape velocity would be harder to reach)”. To exemplify: It would take significantly more time to develop a solution for a large-scale dynamic database than it does a small-scale static web server. Then consider multiplying this up with all the necessary components to create a fully automated environment. Then again; these are just some of the basic elements of the

puzzle one need to assess when deciding if this would pose value to the organization.

6 Regulatory requirements

We live in interesting times and we have all seen the corporate scandals around the world. As a result a number of legislative and regulatory requirements have been and will be imposed on organizations across the world. One of the most debated and intrusive ones are the Sarbanes-Oxley (“SOX”) law that was passed in 2002 in response to a number of major corporate and accounting scandals in the US. As a consequence of these scandals, public trust in the accounting and reporting practices of these and other major corporations were lost.

So what does SOX actually address? In simple terms it establishes the following:

- New standards for corporate boards and audit committees.
- New accountability standards and criminal penalties for corporate management.
- Independence standards for auditors, whom in term have to be external to the organization completing an audit.
- A Public Company Accounting Oversight Board (“PCAOB”) under the Security Exchange Commission (“SEC”); making sure that public accounting firms follows defined legal requirements.

The immediate question would be; “how does this equate to applying application and service provisioning?”. A great deal in fact! Provisioning techniques can assist in underpinning a drive for SOX compliance. Adapting a release-, change- and configuration management process with the associated technical solution, solves a number of the imposed IT ‘*requirements*’ as defined within the SOX act. In simple terms; provisioning techniques deliver an ‘*engine*’ for which accountability can be delivered to the business from IT. A practical example is to ensure that only authorized personnel can provision out applications, services and any change to them, into the disparate environments. This effectively ensures that all the steps are audited and historical records of what has been done (and by whom) are being kept. This is key since these are required ‘*control measures*’ inherently required within SOX; effectively ensuring that in the end financial ‘*data*’ (whatever that may be) cannot be tampered with.

7 Conclusions

Applying an automated provisioning platform (or methodology) to a data centre environment will yield substantial benefits to many organizations, both tangible and intangible. To conclude, what are the things to remember? The list beneath summarizes some of them:

- Define your requirements on a process level before looking into tools and technology. They should adapt to the process what works for you; not vice versa!
- Be sure to engage all parties in your organization in the quest to define the processes that work for you. Through this you involve them as stakeholders and will have their support throughout the project. If not, you will fail!
- Never underestimate the people when adopting a provisioning strategy. Process and technology is “simple”, people are not!
- Think long-term; make the provisioning strategy an integral part of your IT operation. It will ensure that you are able to ‘*do more with less*’ over time.

Acknowledgments

We are grateful to acknowledge the support of Gary Kelly and Grant Rickard for providing the opportunity to spend time preparing the manuscript for the paper and conference presentation. We would also thank the following individuals as we would not have been where we are today, without them; Peter Charpentier, Matthias Pfützner, Brett Goshorn, John Stanford, Tim Read, Keith Black, Henning Hogness, Jan Bjelde, Ragnar Hongset, Arne Gigstad and Leif Wold.

Availability

This paper is available through the AUUG 2005 conference as well as through the AUUG media. It will also be available on Jorgen Skogstad’s website; <http://www.skogstad.com/AUUG/>

References

- [1] IT Infrastructure Library (“ITIL”). <http://www.ogc.gov.uk/index.asp?id=2261>
- [2] Control Objectives for Information and related Technology (“Cobit”). <http://www.isaca.org/>
- [3] Sun Microsystems’ N1 technology. <http://www.sun.com/software/n1gridsystem/>

[4] IBM onDemand business. <http://www-306.ibm.com/e-business/ondemand/us/index.html>

[5] EDS' agile enterprise.
<http://www.eds.com/services/agileenterprise/>

[6] CM Crossroads.
<http://www.cmcrossroads.com/>