A Systems Thinking Approach to IT Process Automation
Gaining Efficiencies in Very Large Multi-Service Data Centers

By

Scott E. Albrecht

Bachelor of Science, Information Technology and Business
University of Massachusetts, 2011

Submitted to the Engineering Systems Division in partial fulfillment of the requirements for the degree of

Master of Science in Engineering and Management

At the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

December 2013
[February 2014]

Signature of Author: Signature redacted

Scott E. Albrecht
Engineering Systems Division
December 31, 2013

Certified by: Signature redacted

James M. Utterback, Thesis Supervisor
David J. McGrath Jr. (1959) Professor of Management and Innovation and
Professor of Engineering Systems, MIT Sloan School of Management

Accepted by: Signature redacted

Patrick C. Hale
Director of System Design and Management Program

THIS PAGE INTENTIONALLY LEFT BLANK

A Systems Thinking Approach to IT Process Automation
Gaining Efficiencies in Very Large Multi-Service Data Centers

By

Scott E. Albrecht

Submitted to the Engineering Systems Division On January 13, 2014 in partial fulfillment of the requirements for the degree of Master of Science in Engineering and Management

ABSTRACT

Keeping up with the Joneses, in an Information Technology (IT) sense, is not a feel good activity, it's a necessity to remain competitive. Building and maintaining a relevant, reliable, and scalable IT service infrastructures, without crushing the bottom line, is a necessary undertaking to avoid obsolescence in the marketplace. This is particularly true for very large scale IT Service and "Cloud" providers. At the very top of many CIO's wish list is to obtain, or create, an effective and efficient IT Process Automation (ITPA) framework. Use of ITPA or Run Book automation is a requirement to efficiently manage increasingly massive pools of systems and services under any particular IT Service Provider's management domain. A successful process workflow, run book, automation, and orchestration framework implementation requires a high degree of flexibility and scalability to be successful. It also requires an intuitive command and control structure to manage today's massive scale deployments and their increasingly demanding customers and service level agreements. This paper explores a new applications of a "publish-subscribe" messaging paradigm and how it can be leveraged to construct a core ITPA framework. This ITPA framework will scale to match the various needs of very large IT service infrastructures. The overarching intent of the paper is to discuss this ITPA framework, at a level of detail sufficient enough to provide a well-trained IT practitioner the ability to construct it themselves within their organization. This paper is however abstract enough to give the practitioner a high degree of choice with regards to the specific technologies and implementation details that must ultimately be tailored to their organization's specific needs and requirements.

Thesis Advisor: James M. Utterback

Title: David J. McGrath Jr. (1959) Professor of Management and Innovation and Professor of Engineering Systems MIT Sloan School of Management

ACKNOWLEDGEMENTS

I am extremely grateful to have had the opportunity to work with such a talented and experienced group of students, professors, and supporting staff. The experience while MIT was encompassing and immersive. It has become part of me and I will carry it forever.

I would like to extend a special thank you to Professor Jim Utterback. Professor Utterback was not just influential in the construction of this paper but was also a very important pillar to my entire experience at MIT. He has helped me understand the broader context of the technology that I spend my day to day immersed in and introduced me to a cast of brilliant individuals whose work has helped to recalibrate the way I think about Business and Technology. Professor Utterback has also given me the freedom and confidence to challenge conventional wisdom and think on my own, particularly with regard to Technical Strategy.

Pat Hale and the entire System Design and Management staff are so friendly and helpful in removing obstacles for us. This enables us the students, the ability to focus on our core educational mission. The whole team is critical to the success of the program and their fine work does not go unnoticed, and so, deserves special thanks.

To my management at my employer Oracle for giving me the assistance, flexibility, and balance necessary to make my school, work, and life all complement each other.

To my wife and children for giving me the support, time, and encouragement to continue my studies. They bring purpose to my life.

TABLE OF CONTENTS

Abstract 3

Acknowledgements 4

Introduction 9

Chapter One – The IT Service Provider 13

A Management View of the IT Service Provider 13

A Systems Level View of the IT Service Provider 16

Chapter Two - IT Process Automation and Gaining Efficiencies 26

Run book and IT Process Automation 27

Approaches to Automation 31

Orchestration and Automation Products 38

Building a Messaging Framework through Publish-Subscribe and Broadcast Paradigms 41

Addressing the Systems in Fleet with Attribute-Value Pairs 45

Automation via Horizontal "Group Based" Orchestration 49

Automation via a Micro-Orchestrator 56

Automation via a Macro-Orchestrator 60

Chapter Three – Design Considerations and Best Practices 64

Understanding Safety and Managing Change through ITPA 64

Pitfalls and Failures of Automation 71

Conclusions and Future Work 74

References 75

TABLE OF FIGURES

Figure 1 - The IT Automation Closed Loop (Adapted from Source: Forrester Research, Inc.) (Garbani, Mendel, & Radcliffe, The IT Automation Imperative - Putting IT On The Road to Industrial Mass Production REPORT -, 2009) 11
Figure 2 - Logical Data Center 17
Figure 3 - Generic Service to Abstract Architecture Mapping 18
Figure 4 - Types of Service Tenancy 19
Figure 5 - Complexity Model for Homogeneous Services 21
Figure 6 - Large Heterogeneous Multi-Service Datacenter Model 22
Figure 7 – Survey - What are (or would be) your primary reasons for adopting or considering IT process automation? (Enter top three in priority order) 29
Figure 8 – Survey - Which disciplines have you already automated (or are in the process of automating) (Enter all that apply) 29
Figure 9 – Survey - What is your highest-priority focus for IT operations process automation for the next 12 to 18 months? (Select 1) 30
Figure 10 – Survey - How would you assess you IT operations process level? (Select 1) 30
Figure 11 - Two Process - Single System Workflow 33
Figure 12 - Two Process - Two System Workflow 35
Figure 13 - Seven Process - Three System ITPA Workflow – The External Orchestrator 37
Figure 14 - ITPA - RBA Gartner Vendor Landscape 38
Figure 15 - Transitioning View of the Operating System 42
Figure 16 - Publish-Subscribe Message Flow Example 43
Figure 17 - "Pseudo" Message 45
Figure 18 - Forming Groups through Attributes 49
Figure 19 - Horizontal Group Based Orchestration 50
Figure 20 - Group Based Micro-Orchestrator 57
Figure 21 - The "Micro Orchestrator" 58
Figure 22 - The Macro Orchestrator - High Level Design 61
Figure 23 - The "Macro Orchestrator" 62
Figure 24 - Relationship between Change, Growth, and Obsolescence 65

Figure 25 - Possible "Safety Catch" implementation 67
Figure 26 - Single Command Repository vs Synchronized Command Repository 69

TABLE LIST

Table 1- Key Statistics of Data Processing & Hosting Services in the US 2013 (Recreated from IBIS World Report) (Kraveepetcharat, 2013) 14
Table 2 - ITPA Criteria to Benefits Mapping 24
Table 3 - Top Three IT Automation Priorities Survey (Williams, 2010) 28
Table 4 - Possible Attributes Associated with a Rock Star's Fan 47
Table 5 - Possible Attributes Associated with a System 47
Table 6 – Several Other Possible Attributes Associated with a System 48
Table 7 - Forming Groups of Systems in the System Fleet 48
Table 8 - Case 2 Operational Message Sequence 53
Table 9 - Automation Design Factors that Lead to Difficulty 73

THIS PAGE INTENTIONALLY LEFT BLANK

INTRODUCTION

Information Technology organizations throughout the world are constantly adapting to a more demanding IT consumer; business or otherwise. Over time, this consumer has grown increasingly less tolerant of application downtime, system performance degradation, data loss, security failures, or long cycle times to add new features to a given service or application. This is especially true given that IT "services", now more than ever, play an essential role in the core operations of businesses of almost any size. These service demands are also not limited to only core business functions like employee collaboration suites, human resource, finance, and customer relationship management (CRM). Even the day to day consumer of internet based "retail" applications also demand "up 24x7" and even for "free" services. The types of free services may include Facebook, Twitter, Google services, Yahoo, Skype and an endless array of others. These services provide everything from your individual bank account management to customizable weight loss programs. The expectations on these free services are particularly enormous. A zero downtime and defect policy is not a stretch goal, it's the baseline expectation. Additionally, these service demands are not only placed on IT staff and management, but also on their counterparts in the applications development side.

Occurring in parallel to the ever increasing service demands are the mounting complexities of the actual applications that are built and maintained IT organizations. As an example; for every application service that has become a ubiquitous commodity, there is the inevitable modularization, consolidation, and integration of these commodities into the next great "suite" of products or platform services. These suites and platforms will ultimately go through the same commoditization, modularization, consolidation, and integration until they themselves become ubiquitous. This cycle makes way for the next great platform or service. This full lifecycle seems to have accelerated recently with the emergence of "cloud" based applications suites, services, and platforms. The expectation from the consumer's perspective is instantaneous, persistent, and stable availability of highly complex and customized application service environments. Just a few short years ago for an IT staff to bring to life an identical suite of would require a massive systems engineering effort and significant upfront capital. Now a CTO can "order up" this same service from a cloud based provider in approximately the amount of time it takes to complete the registration process. Then like magic it appears! But just like any good magic there are two sides to the equation; the magician's side and observer's side. What the observer sees is only the end result of the

magician's preparation and practice. What the observer does not see are the weeks, the months, or the years spent in preparation by the magician.

With that said; this paper is predominantly written for the architect or practitioner who provides and builds these mega-services for the consumer. A main focus of this paper will be on spent on understanding the IT processes and process automations that an IT organization must develop and maintain in order to efficiently produce these services and applications with any sort of scale.

As an example of automation; 5-7 years ago it might have taken an IT staff of 20 well trained system administrators, database administrators, developers, architects, and managers anywhere from 3-6 months to manually install, customize, and deploy a large enterprise application for a given enterprise customer. Now it is possible to have and equivalent application ready to populate with custom data in less than a day, and sometimes instantly. This act will be done with virtually no human intervention on the construction side. This system is bought up wholly through automation. Specifically through an automated IT provisioning processes. This same automated provisioning process has the capability to create any number of similar systems for any number of enterprise customers. These capabilities are bounded only by the available infrastructure resources and system capacity. Previously the limitations might likely occur as a result of insufficient staffing or personnel to create and manage a given service infrastructure

IT Process Automation (ITPA) does not end after provisioning either, it persists in all aspects of the service's application lifecycle. These aspects of automation lifecycle likely mirror the typical aspects of a single application's lifecycle including; provisioning, change management, release management, performance management, disaster recovery, support, and decommissioning.

When an IT organization is tasked with maintaining the lifecycle of a single enterprise application the need for automation can be somewhat muted. Typically the necessary tasks of maintenance, support, upgrade and performance management can be handled somewhat manually by the existing IT staff. However when an IT organization is tasked with maintaining several thousand enterprise applications, the need for process automation is unquestionably essential. In a recent Forrester article titled "The IT Automation Imperative" the author states the following: "With 35% of a typical IT budget spent on personnel, staffing represents the heaviest IT financial burden. The way in which IT evolves in complexity and size, combined with the current inefficiency of many IT management processes, leads to an ever-increasing demand for additional personnel to simply maintain the productivity status quo. In any case even if IT could hire extra

resources, there is a general shortage of experienced personnel that would preclude organizations from doing so....IT has to embrace the key elements of mass production and adopt automation." (Garbani, Mendel, & Radcliffe, The IT Automation Imperative - Putting IT On The Road to Industrial Mass Production REPORT -, 2009)

Process automation is not always a good thing, however, particularly poorly designed or "Clumsy Automation" (Billings, 1997) can often times lead serious unforeseen negative consequences. Overreliance on automation can have the effect of degrading the necessary skills required to troubleshoot the system or systems when things do not go according to the automation plan. Moreover automation can have the effect of masking to a high degree the complexity of a given IT process.

Given these facts, many of the automation techniques shown in this paper have consideration that both humans and machines will continue to remain vital controllers in the IT process automation loop shown below:

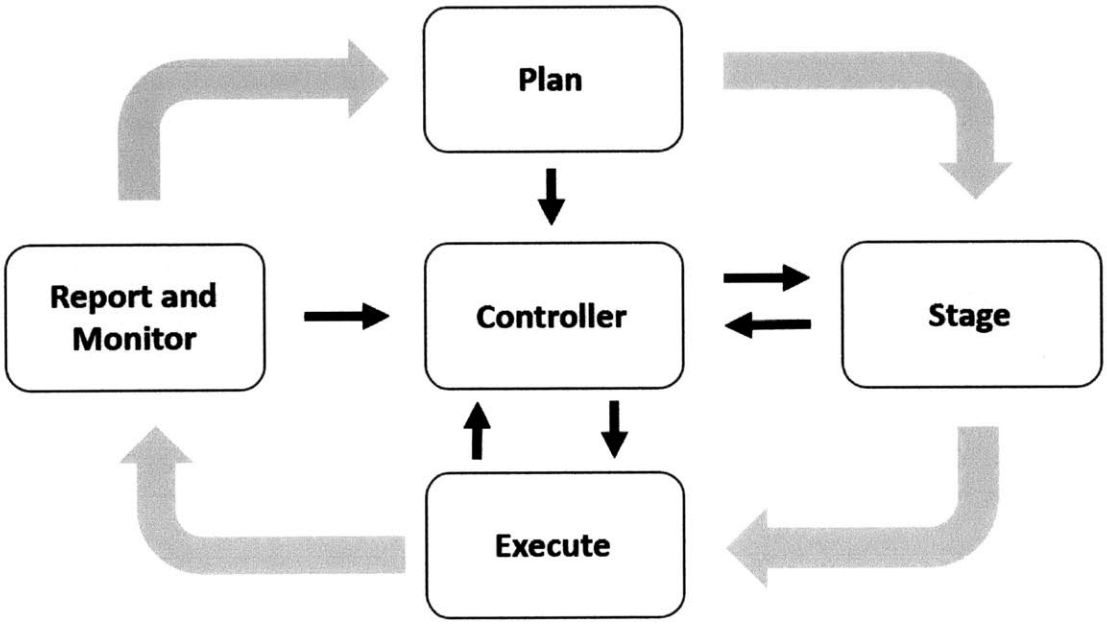


Figure 1 - The IT Automation Closed Loop (Adapted from Source: Forrester Research, Inc.) (Garbani, Mendel, & Radcliffe, The IT Automation Imperative - Putting IT On The Road to Industrial Mass Production REPORT -, 2009)

Additionally, even with extremely well designed and implemented IT process automations, the laws of entropy through time will generally create a more disordered system and therefore increase the probability of unforeseen process errors. These will emerge from several facets of the system and control structures that may not have been available as a consideration to the IT process architect or automation

designer. This paper will try to give some research and experience based approaches to the design of a good IT Process Automation framework that will include the ability to have both human and machine based system controllers and orchestrators.

The flow of this paper is summarized below:

Chapter 1 - The IT Service Provider: *An introduction to IT Service Providers, who they are, and what they do. This Chapter provides a managerial and systems level view of the services and system architectures prevalent today. This chapter also describes some common design patterns used within the data-service center that an IT Service Provider will leverage to provide services to their customers. This chapter also illustrates the operational and systems management needs that must be satisfied in order to maintain demanding service level agreements with their customers.*

Chapter 2 IT Process Automation and Gaining Efficiencies: This chapter d*escribes IT Process and Runbook Automation (ITPA), what it is, and why it is one of the top priorities of any modern IT organization today. The reasons for this are even more pronounced for large scale IT Service Providers; e.g. those generating and providing "cloud" or "as a Service" based products. This chapter illustrates a unique framework, which recombines several existing technologies in a new way. This new framework provides a core ITPA backbone to any IT Service organization wishing to enable any variety of "scaled" process automation and orchestration to fit their particular needs. This framework can be applied to virtually any sized IT Service provider with service infrastructures of any almost degree of complexity.*

Chapter 3 – Design Considerations and Best Practices: *Gives some helpful tips, insights, and recommendations that should be used when designing an IT Process Automation scheme for your organization. The topics include Safety, Change Management, Standardization, Performance, and Configuration Management. These topics are interrelated as design considerations that must be though through when implementing and building out the specific flows enabled by the ITPA framework discussed in Chapter 2, and more broadly any process or workflow automation scheme.*

CHAPTER ONE – THE IT SERVICE PROVIDER

What is an IT Service Provider, where do they come from, and what do they do? Loosely, the term Service Provider describes a company or entity that provides traditional IT services in an outsourced way to another company, entity, or consumer. The Service Providers focused on in this paper will typically be External Service Providers who provide a sizable level or number of services to a large consumer base. The contents of the paper does not only apply to External IT Service Providers however, as many of the suggestions, approaches, and architectures are equally applicable to all IT Service organizations of any type or size.

The Information Technology Infrastructure Library - (ITIL) Glossary defines an External Service Provider as: "An IT Service Provider which is part of a different organization [than] their customer. An IT Service Provider may have both internal customers and external customers. [Where] An IT Service is defined as: A Service provided to one or more customers by an IT Service Provider. An IT Service is based on the use of Information Technology and supports the Customer's Business Processes. An IT Service is made from a combination of people, processes, and technology and should be defined in a Service Level Agreement." (Rance, Stuart; Hanna, Ashley; Hewlett-Packard, 2007)

A Management View of the IT Service Provider

This section will review the Service Provider Industry as a whole, and describe what they do, what their current growth patterns look like.

For the most part the IT Service Provider industry is inclusive of, but not limited to, the following generic and conceptual Information Technology categories:

- *Application and Business Process Outsourcing*
- *Cloud Computing (PaaS, SaaS, IaaS, *aaS)*
- *Infrastructure Outsourcing*
- *Web Hosting Services*
- *IT Utility Services*

These generic categories take many specific forms. For example, a "PaaS" Service Provider could provide any number of services including, application server functionality, raw database services, or any facilities that may be used in the creation of a fully functional application. This contrasted against "SaaS" where the application functionality is already build shows some distinctions within the "Cloud Computing" category. In general there are spanning and somewhat deep hierarchies amongst each category.

The IT Service Provider industry is quite large and according to a July 2013 report by IBIS World the Data Processing & Hosting Services Industry (The IT Service Provider Industry) is approximately $84 Billion in Just the US alone. (Kraveepetcharat, 2013) The worldwide market is certainly larger but the trends that run through the IT Service Providers industry in the US can be used as a reasonable sampling of how the remainder of the markets worldwide are moving.

The following are several statistics about the US IT Service Provider Industry. These statistics are helpful in revealing some fairly interesting trends:

Year	Revenue ($m)	IVA ($m)	Establishments (Units)	Enterprises (Units)	Employment (Units)	Wages ($m)	Corporate Profit ($b)
2004	60,111.80	32,223.70	46,987	40,804	264,980	17,556.40	1,246.90
2005	62,932.70	32,542.60	48,038	41,224	265,260	17,187.00	1,456.10
2006	67,036.80	33,628.30	52,000	44,149	264,320	17,271.40	1,608.30
2007	73,667.30	35,879.20	57,628	49,324	266,130	17,904.30	1,510.60
2008	74,520.80	36,024.10	58,190	49,800	263,140	17,841.00	1,248.40
2009	76,758.40	36,102.50	58,082	49,721	254,140	17,373.40	1,362.00
2010	77,062.80	35,408.60	57,829	49,522	239,850	16,605.30	1,800.10
2011	78,277.30	36,362.60	55,587	47,529	245,573	17,263.00	1,928.00
2012	82,638.50	35,626.10	52,746	47,239	215,797	15,462.30	1,942.50
2013	83,845.50	35,871.40	50,620	47,744	214,404	15,413.10	1,960.10
2014	86,312.30	36,083.20	45,747	44,789	204,318	15,023.00	2,031.80
2015	89,768.30	36,984.40	43,362	42,621	199,729	15,080.90	2,134.90
2016	89,930.30	37,184.20	41,463	40,916	196,393	15,241.20	2,221.20
2017	94,451.20	38,479.30	40,082	39,847	192,978	15,433.20	2,275.60
2018	97,336.70	39,229.70	38,750	38,514	188,712	15,479.50	2,446.10

Table 1- Key Statistics of Data Processing & Hosting Services in the US 2013 (Recreated from IBIS World Report) (Kraveepetcharat, 2013)

One can easily spot that revenue, corporate profits, and the Industry Value Added (IVA) for the Service Provider Industry in the US are increasing at a reasonable clip annually. In the same table it is also quickly noticeable that the industry is expected to continue reducing the number of establishments, enterprises and employment. At first glance it appears from the data that wages are somewhat stagnate, however given that the total number of employees working in the industry are expected to decline when combined with the fact that total wages are expected to remain the same implies a healthy 15% projected average bump in per employee wages over the next 5 years for those that remain in industry.

IBIS provides the following insights; "The Data Processing and Hosting Services industry provides infrastructure for hosting or data processing services used for a variety of information technology (IT)-related activities, ranging from web hosting to automated data entry services. During the five years to 2013, businesses have increasingly outsourced their IT infrastructure needs, directly benefiting industry

operators. Therefore, the industry has fared well over the period, with revenue growing at an estimated annualized rate of 2.4% to $83.8 billion, including expected growth of 1.5% during 2013.

Investment in outsourcing application hosting to specialized firms has been the primary driver of growth, as it has been an alternative to local hosting of enterprise software. Because revenue depends on subscriptions, the pullback of IT spending during the recession [of 2008] slowed the rate of revenue growth. However, spending has picked up since 2011, and it is expected to grow even faster in 2013 as firms increasingly outsource their IT needs to third parties.

Mergers and acquisitions are anticipated to increase during the next five years as firms consolidate to increase the subscriber base over which they can allocate computing resources. Also, supply disruptions in the hardware space (such as the flooding of hard-drive manufacturing plants in Thailand during 2011) may push companies that managed their IT infrastructure needs in-house to opt for a third-party provider. Other factors will also play into growth in coming years. Consolidation within other industries will push businesses to outsource their IT needs as systems become too complex to maintain in-house. At the same time, the exponential growth in complexity as scale grows will increase the expertise level needed to effectively manage large data centers. As firms begin capturing more data, they will increasingly require outside expertise to manage their data collection, hosting and processing. As a result, revenue is forecast to grow at an average annual rate of 3.0% during the next five years, reaching $97.3 billion in 2018." (Kraveepetcharat, 2013)

The global marketing push toward "cloud" or "*aaS" ([Anything] as a Service) based services are wrapped into the broader category of IT Service Providers. Though the "*aaS" providers do not comprise the full spectrum of IT Service Providers they do represent a very large section of this market, at least in name.

There are roughly three main categories of AaaS Providers in this industry. Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). These trends have forced many commercial off the shelf (COTS) vendors in the software industry to rapidly develop and implement a "cloud" or "as a Service" based strategy at least around their core software components. This conversion is at a minimum disruptive. The trend forces software firms to shift from a central role and mindset of development into to one of operations management. In this new role characteristics like uptime become a key part of corporate strategy as opposed to a recommended best practice in the deployment of their applications or application suites. In this type of environment the COTS vendor becomes responsible for

far more than delivering the software packages, instruction manuals, and consulting services. They now become responsible for managing the availability and vigor of the deployed and in production application.

A Systems Level View of the IT Service Provider

By and large the Service Provider Industry makes use of very large data centers to deliver services to their consumers over public networks. These data centers are leased or owned (in part or in whole) by a given IT Service Provider. The data centers are typically utilized to provide the "**ping** (Remote Management), **power** (Electrical Power), **pipe** (Internet Connectivity)" necessary to run a set of servers and equipment required to deliver the applications and services to their consumers. The view of the data center for the purposes of this paper is logical rather than physical as a system's location within the "network" will to be more significant to the discussion than the system's physical location within the data center. Also for the purpose of this paper data centers are assumed to provide essential elements such as; power, cooling, ventilation, rack space, modular design, and a fault tolerant electrical and inbound network infrastructure at a "Tier 4" level of service. Tier 4 is a "standard" level set forth by the Uptime Institute's Data Center Site Infrastructure Tier Standard. (Uptime Institute, LLC, 2010-2013) The standard set forth by the uptime institute defines, levels of training of staff, maintenance, operating conditions, and locations, all of which provide a robust and resilient data center that is used to provide consistent power and external "ping" to the servers and equipment being hosted and managed.

There is a clear demarcation between the data center and the servers and equipment housed within. For example the data center may remain fully operational whilst the servers and equipment housed within are down and unavailable. An analogy that can be used is that a server or a piece of equipment is like an airplane and the data center, with its core services, is the airport. One or many individual airplanes can be down for maintenance or repair at any moment while others remain taxiing, taking off, landing, or parked under normal operation. An airplane's actual state has no impact on the overall state of the airport. In much the same way, one or many servers, network appliances, or infrastructure components (equipment), may be down for a period of time, but their outage will not impact the availability of core data center services of the data center itself.

In general the core operations and maintenance *of* the data center itself will be out of scope for this paper. However the operations and maintenance of the servers and equipment *within* the data center are in scope. See the boundary diagram below in Figure 2 below.

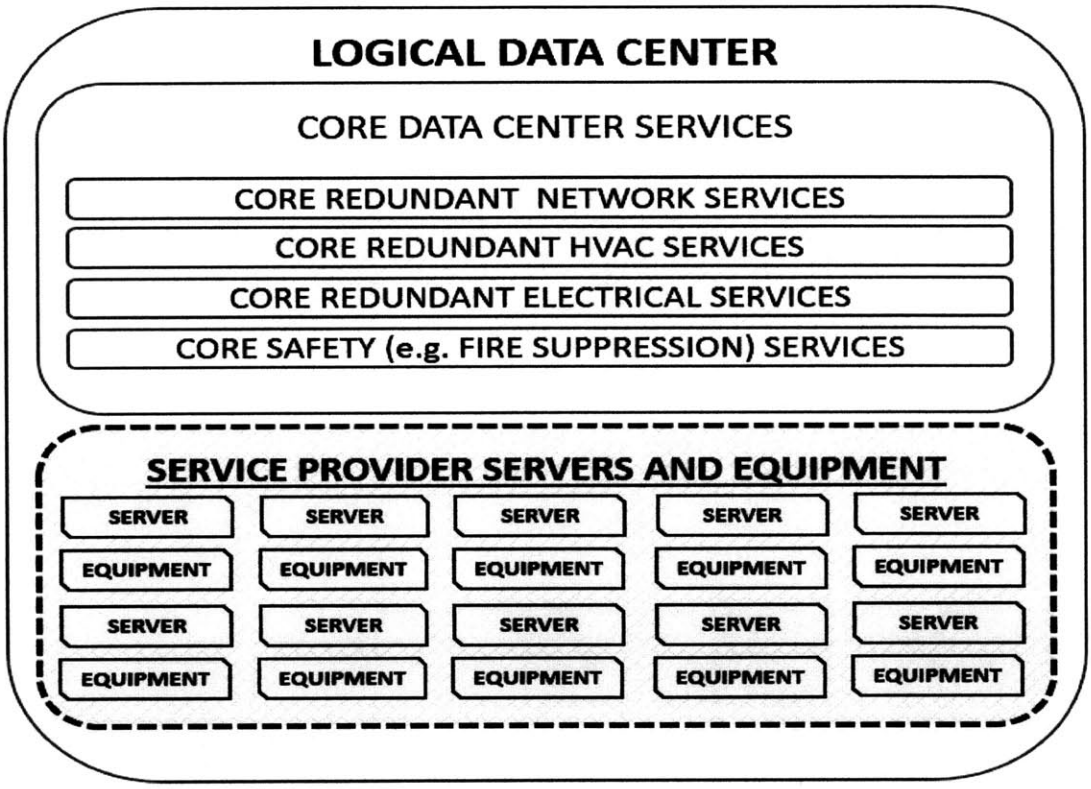


Figure 2 - Logical Data Center

This diagram shows this distinction between the core services offered by the datacenter, and the servers and equipment that operate the services for the IT Service Provider. (*Within the dashed line.*) It is not uncommon that an IT Service Provider will have ownership of both the datacenter and the servers and equipment, but they will typically be managed and operated under distinct operational teams with varying roles, processes, and procedures. Within this paper the "servers" and "equipment" shown below within the boundary are in scope, because they are an integral and high touch component of the "service" being provided by the IT Service Provider whereas the plumbing and availability of the "core data center services" are assumed to always be available and therefore somewhat ubiquitous.

Given the fact that the "servers" and "equipment" above *are* in scope, a discussion about the taxonomy of these systems, as they relate to types of services provided, will be discussed further in this paper. For now however, it should be understood that the servers and equipment are simply the devices that provide the raw computing resources and pathways to enable a given service. Specifically the CPU, RAM, storage, and local network resources.

A particular service can have partial or whole ownership of a system or group of systems and equipment. In fact, the ways in which services utilize servers and equipment will vary extensively from one Service Provider to the next. These variances are generally limited only to the imagination of the infrastructure and application architects. Many of these service architectures (though perfectly valid) may look nothing alike from one Service Provider to the next. This despite the fact that the services provided may be almost identical. There are some common architectural themes or patterns that exist with regards to the services being provided. These services are then mapped to these architectural pattern, and these architectures make use of the servers and equipment as shown below. Again, the mapping is somewhat arbitrary, and the list is limited to provide a few simple examples.

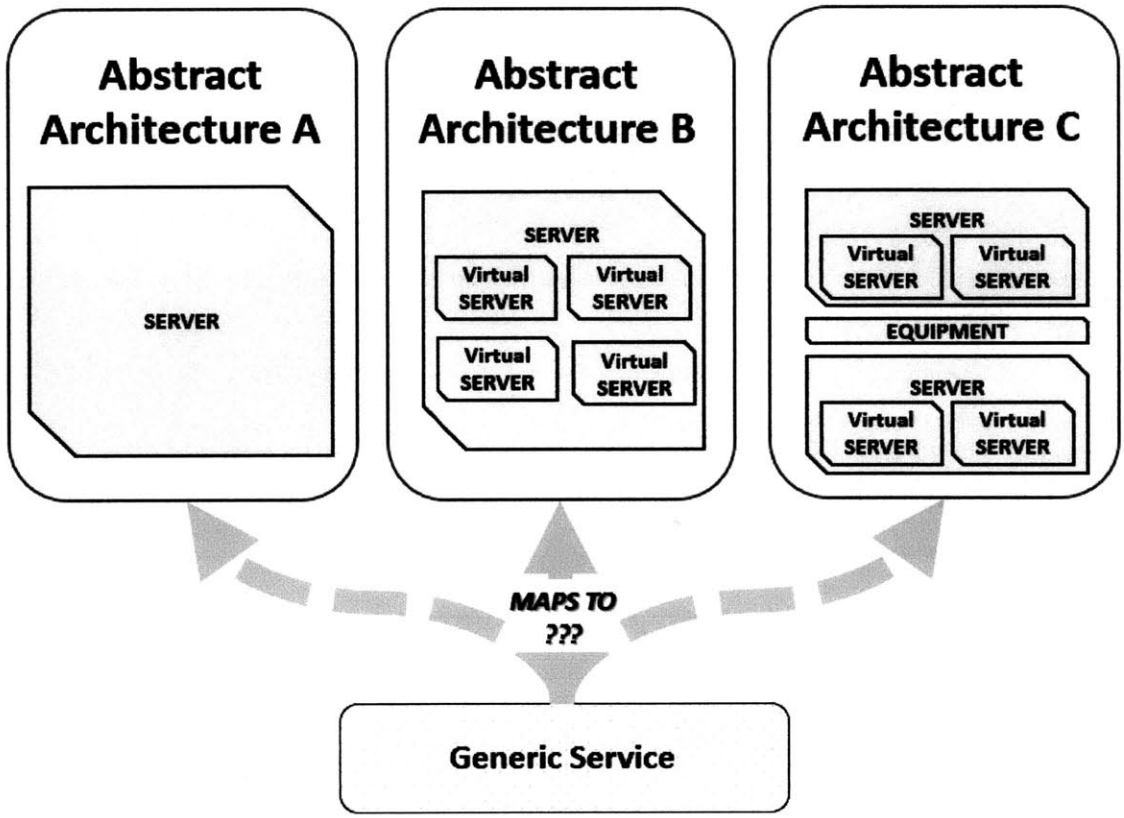


Figure 3 - Generic Service to Abstract Architecture Mapping

Modern system architectures involve "bare-metal" systems and virtualized systems. Both types can be either single or multi-tenant. The diagram above shows how a single "Generic Service" could run from Architecture A, Architecture B, or Architecture C. This type of service mapping is defined as a "single-tenant" type solution. This mapping does not need to be 1:1 however. For example, it is entirely possible

to have many services mapped to a single architecture. (e.g. : map 5000 homogeneous services onto a single Abstract Architecture A, B, or C) In doing so the consumers of the services would "share" the resources used by the service but the sharing would be somewhat abstracted to the consumer of the service as their view would be one of isolation if done properly. This is a type of Multi Tenancy which has become pertinent with the widespread adoption of "virtualization" technologies by IT Service Providers.

In Gartner Reference Model for Elasticity and Multitenancy (Natis, 2012)they define several models of service "tenancy". Gartner states that ***multitenancy*** is the "Sharing common application computing resources among independent users or processes (tenants) is an essential characteristic of cloud computing, and is referred to as "Multitenancy" (multiple tenants sharing common physical computing resources, while remaining logically isolated). Depending on the usage scenario, tenants may be user organizations or applications; in either scenario, the application instances are running in a physical computing environment partly shared among them, while presenting a logical computing environment [i.e.: a service] that appears exclusively dedicated to each instance (tenant)."

Below is a pictorial description of the "tenancy models" described by Gartner (Natis, 2012).

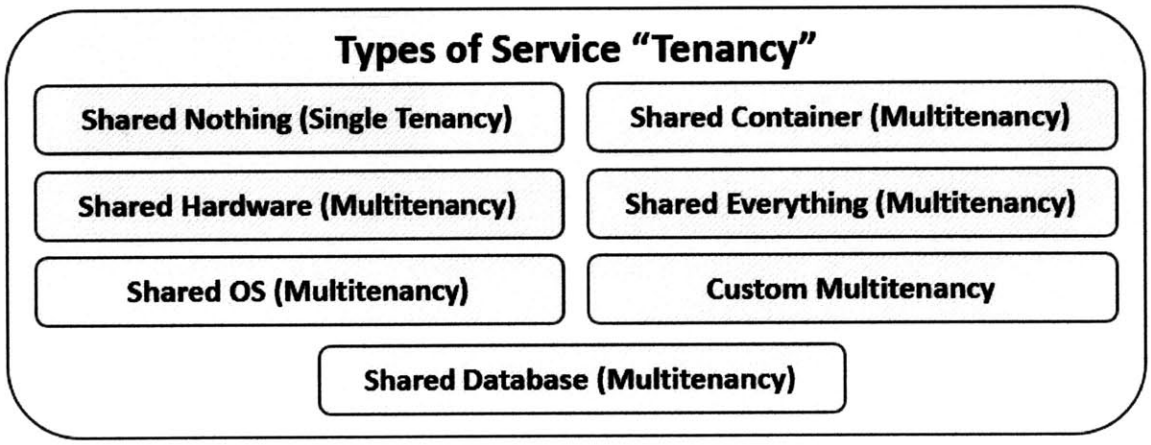


Figure 4 - Types of Service Tenancy

Several implementations of the varying and common types of service architectures and tenancy models are discussed in further detail in the following chapter, but they are referenced here to make the point that there is no single and set way to map services to a given set of servers or equipment. There is also no single or set way to model service tenancy without having a full understanding of the service being offered. Clearly the mapping or deployment strategies will ultimately be customized to a high degree by the needs and requirements of the consumers of a given service. For example, one would approach the

architecture of a service that provides a single tenant bare metal web server (such as an apache HTTP server) to an array of individual customers far differently than an architecture which provides a multi system, multi-tenant ERP service to a set of Fortune 500 companies. Though this paper will not be about the architectures of servers it is important that the idiosyncrasies of these architectures be discussed in some level of detail to understand how to create efficiencies through the various automation paradigms discussed in this paper.

Moving from the discussion of tenancy and architecture comes the topic of complexity of the services offered by the Service Provider. For example, it is not a fair comparison to say that a "service" providing 1000 unpopulated webserver instances to a consumer is equal in complexity to a "service" that provides 1000 Enterprise Resource Planning (ERP) instances. They are vastly different in their deployment architectures, tenancy models, and also in their complexity at an atomic "service" level.

Complexity can go in two directions "vertically" and "horizontally." Complexity can also go in both directions at the same time (Both vertically and horizontally complex). An example of a "vertically" complex service offered by a Service Provider could be something like a CRM, ERP, Supply Chain, or Salesforce automation Suite. A single service can span many physical or virtual systems and have hundreds or even thousands of integration points. An example of a "horizontally" complex service might be a very simple service architecture or design, with very few integration points and system modules, but the complexity will come from how it scales. For example the complexity of running a ten thousand homogeneous webservers is equally complex but in a different way. Perhaps a good metaphor to describe the distinction between vertical and horizontal complexity is to describe the difference between a custom Lamborghini and the 2012 model year Toyota Corolla. The Custom Lamborghini is precision tuned machine, with every component calibrated and tuned to an exact specification for that single car. However, when you look at the "Just in Time" production line from which the Toyota Corolla is produced one can easily see how immensely complex the production of the corolla is despite the fact that the complexity of a single Corolla pales in comparison to the complexity of a single Lamborghini. Below is an example of the complexity models described above:

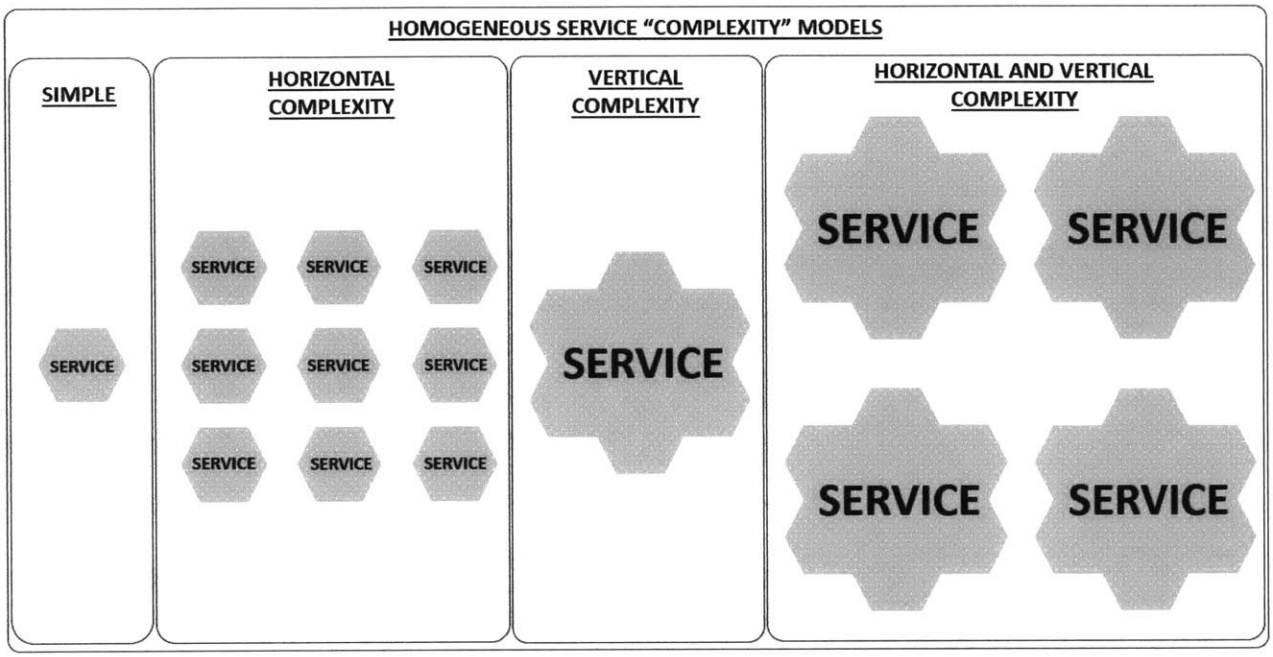


Figure 5 - Complexity Model for Homogeneous Services

Each hexagon shown above represents a single abstracted "unit" of complexity as it relates to an autonomous service. Because this is an abstracted measure of complexity, a single unit could represent many things. For example; a service with no integration points that performs a limited number of functions might have only a single "unit" of complexity. On the hand, if there is a vertically complex service it might contain many "units" of complexity within a single service as shown in the "Vertical Complexity" model above.

In the Horizontal Complexity model there are many autonomous services with limited complexity that exist under a given management domain. Despite the fact that any single service might not be complex in and of itself, the management of hundreds or thousands of these autonomous services will ultimately create a high degree "horizontal" complexity from its scale. As the number of autonomous services increases so will the complexity of creation, management, and maintenance these services. Imagine 200,000 homogeneous webserver instances.

In the Vertical Complexity model above, the more complexity units a single autonomous service has, the more complex the single autonomous service will be. A service with a high degree of vertical complexity might have dozens of systems, hundreds of integration points, and perform any variety of complex functions. In much the same way as Horizontal Complexity is positively correlated to the Service Provider's

management complexity, so is Vertical Complexity. The more vertically complex a service is, the more complex it will be to create, manage, and maintain it by the IT Service Provider.

In the case of most Service Providers the services offered are both vertically and horizontally complex and generally not homogeneous. The will also have many types of tenancies (described above) to formulate the service architecture. As you can imagine a typical Service Provider's datacenter can become incredibly complex from both a systems and system's management perspective.

Below is an example of how a very large heterogeneous, multi-service data center could look using our abbreviated complexity model.

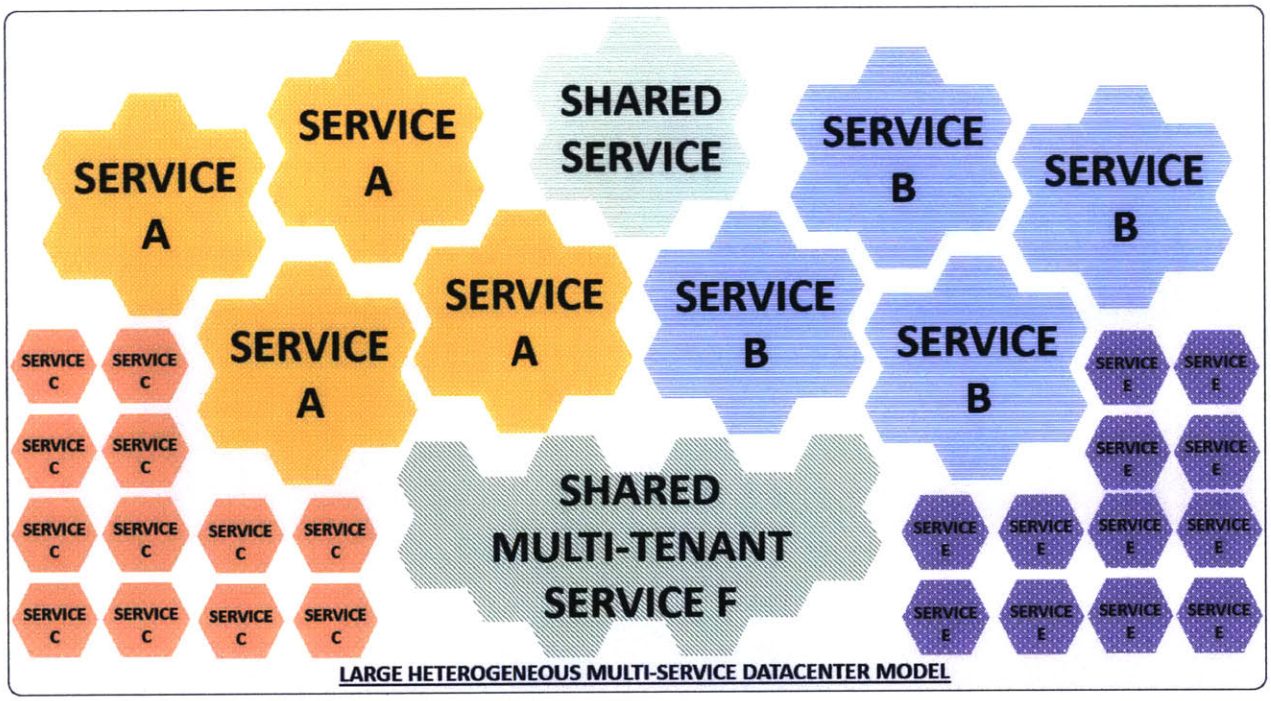


Figure 6 - Large Heterogeneous Multi-Service Datacenter Model

The model above illustrates one of the key complexities that any modern Service Provider must address from a systems perspective. In order for an IT Service Provider to manage this incredible degree of complexity it must find efficiencies in almost every aspect of the operational lifecycle. To gain efficiencies in managing these systems the Service Provider will undoubtedly need to highly automate if they expect to compete.

Thought IT Process Automation will be discussed in detail in further sections of the paper we will introduce some of the key pillars that can fall under the general umbrella of IT Process Automation. According to a recent Forrester Market Overview of IT Process Automation (Garbani & O'Donnell, Market Overview: IT

Process Automation, Q3 2011, 2011)the following four "solutions" represent the ***Foundation of IT Process Automation***:

Workload automation - This is the earliest and most common form of IT process automation. Any IT organization that reaches a certain level of complexity needs a workload automation solution to insure timely and accurate performance of its asynchronous processes.

Run book automation - This is the second wave of IT process automation, born from the complexity and resource-consuming activities of IT operations. Server provisioning is the most widely adopted use for run book automation, but the principles are now applied to virtualization control and the diverse operations needed to make applications cloud-ready.

Low-level process automation - This type of automation replaces script-based IT processes for typical IT operation activities, such as implementing a patch on an operational server. Other forms of low-level automation are those used to integrate applications at the user interface level

High-level process automation - Release management, change control, and complex ITIL derived processes are all examples of high-level IT processes that can be automated. These workflows often reside within a service desk engine, but this is certainly not a requirement.

Complementary Technologies and Capabilities (Garbani & O'Donnell, Market Overview: IT Process Automation, Q3 2011, 2011)

Integration into a single automation framework or solution - These solutions have been brought under a single umbrella and can share data and other interfaces through a common access framework, even though they may not be fully integrated at a process modeling and data modeling level.

Ability to combine automation solutions into a single process - solutions have been brought together as a single and seamless process approach, fully abstracting the differences between automation solutions. The modeling basis is unified either via a fully-shared model or via an abstraction layer atop disparate models, which offers the appearance of a common model.

Automation self-service - This is the first stage where IT process automation meets business process automation. The end users of the process can choose from a service catalog and initiate the process execution themselves — without the need to involve someone from IT operations. At this stage, automation presents either complete process suites or expose reusable components

that can be stitched together into new processes. Eventually, end users can plan execution of business process steps that will self-provision resources and load applications to achieve business objectives.

Ability to use SLAs or history to trigger automated processes - Behavioral insight such as performance, availability, and capacity factors governs the execution of any process by analyzing service behaviors and then triggering the appropriate actions. These decision triggers are most effectively tied to service objectives defined in service-level agreements (SLAs). This capability is used to plan execution resources, either based on a historical perspective (for example resource usage for this job at this time of the year), resource forecast (based on capacity management), or any form of predictive analysis that will help provision resources automatically.

Integration with IT management solutions - Using SLA or any performance-oriented data requires a close integration with IT system monitoring. This could be achieved by integration with existing solutions or by including a monitoring capability within the automation solution. The latter is far less common, but fuller automation suites with such capability are emerging.

Use of complex event processing as an automation trigger - As performance data and other events become available, an analysis of these events coming from multiple sources can in turn create an event that will trigger an automated operation. Ideally, CEP will combine event sources from the business side and from the IT side to reach a conclusion and launch an operation.

The following table from Forrester (Garbani, Mendel, & Radcliffe, The IT Automation Imperative - Putting IT On The Road to Industrial Mass Production REPORT -, 2009) correlates these criteria of ITPA to the Benefit that will be derived by a given IT Service Provider:

Table 2 - ITPA Criteria to Benefits Mapping

ITPA Criteria	Benefit to IT Operations
Workload Automation	Automated execution of asynchronous application processes, based on date/time, events, or user request
Run Book Automation	Generic form of automation through the generation of scripts based on a library of potential operations conducted on software launched in context
Low – Level Process Automation	A form of run book automation that triggers processes used in IT operations to automate routine technical tasks

High – Level Process Automation	A form of run book automation that triggers processes that can be used in IT operations or in business processes to automate complex workflows.
Integration into a single automation framework or solution.	The grouping of all forms of automation within a single framework with a common a common core technology, and an integrated way of nesting automated processes and a common user interface.
Ability to combine automation solutions into a single process.	The ability to launch a form of process automation from another workflow being executed; for example, launching server provisioning from workload automation
Automation self-service	Availability of catalog of automated processes (or a library of pre-defined processes) that can be launched by a user from a serf service interface.
Ability to use SLA or history to trigger automated processes	Availability to compare a process execution time with a predefined SLA and take some form of automated corrective action.
Integration with IT management solutions	The capability to send or receive events or performance information from an enterprise monitoring solution
Use of complex event processing as an automation trigger	The use of an integrated complex event processing solution or a similar form of analytics to trigger the launch of an automated process.

This paper will largely be used to describe an IT Process automation system that is flexible enough to accomplish these goals illustrated above. Because every workflow, every process, and every IT organization is unique we will attempt to build a unified framework that is simple yet abstract enough to enable customization to account for the highly variable ways in which all IT departments create and manage their own processes and workflows.

CHAPTER TWO - IT PROCESS AUTOMATION AND GAINING EFFICIENCIES

This chapter focuses on approaches that IT Service Providers can take to gain enormous efficiencies through IT process automation. Particularly those with very large IT and supporting service infrastructures. The efficiencies gained are critical for any IT Service Provider who intends to scale their services out without also scaling the workload necessary to build and maintain these services. Process automation can be used to create completely or partially automated tasks and workflows in any number of functional areas. These areas include; change management, provisioning, support, configuration management, or virtually any phase of the target service's lifecycle. The framework described in this chapter can be deployed for both highly complex infrastructures and applications of a very large scale. Additionally the automation solutions that are described can handle a very high volume operations.

Though there are many different approaches to IT process automation, a core understanding of the needs and requirements of the implementing IT organization are necessary to the success of the ITPA implementation. This fact is true regardless of the approach to automation taken. In other words, if an organization doesn't understand what is being automated fully and in the proper context of the implementing organization then their chances of success are less than average.

We start this section by investigating some of the top priorities of the IT industry as they relate to automation. These are helpful to understand the needs, priorities, and rational for process automation as a core IT strategy. This data is followed by an investigation of some of methodologies and tools that can take out some of the heavy lifting with the development of an ITPA strategy. Again, it should be clearly understood that the challenges of implementing an ITPA strategy will not be solved by simply installing a commercial "automation" suite or product. But by presenting how some of these "breeds" of suites, products, and frameworks are used in a "solution-neutral" way, one can begin to develop a direction necessary to help their organization define a core ITPA strategy. At the core of any IT Process Automation effort there must also be a high degree of time allocated to understanding full details the processes and the workflows that will be automated. This work is necessary regardless of the product or framework an organization will choose for their individual deployment.

This chapter concludes by providing examples of process automation techniques that have been deployed and shown to work on a very large production infrastructures. These automation techniques have been proven to give enormous efficiencies to the management, support, and core operations teams in several

very large heterogeneous IT service environments. The IT Process Automation framework described in this chapter has provided very real and significant gains in efficiencies. These and other alternative approaches are discussed not to give "the way" to automate workflows and processes, but rather, to give in some level of detail, "one way" for IT organizations seeking an approach to ITPA strategy.

Run book and IT Process Automation

Why does an IT organization need process automation and what does it look like? Runbook Automation or as Gartner redefined it "IT Process Automation" (ITPA) is becoming a critical component of any IT organization's strategic plan. The value that can be found in ITPA is akin to that found in any modern industrialized production plant but at an IT scale. Specifically, an IT organization can make monumental leaps forward in efficiency, cost reduction, as well as error and risk reduction by implementing a well thought ITPA strategy.

In a 2011 report Gartner states that ITPA "promise(s) to mitigate IT operations risk, reduce costs, reduce complexity and increase operational efficiency. ITPA should not be viewed as an add-on to existing tools, and it should not be viewed just another IT management tool. It is a new IT operations management paradigm requiring new skills, organizational design, investment and support, which unless it is architected upfront will result in unrealized value, additional costs and complexity." (Colville, IT Process Automation: Moving From Basics to Best Practices - REPORT - G00214344, 2011) Moreover Forrester echo's these comments by stating; "The tasks of the IT infrastructure and operations (I&O) professional have become increasingly complex and susceptible to human error. This is a direct consequence of the sheer volume and diversity of business services and underlying IT infrastructure components. By masking IT(s) diversity and automating highly repetitive tasks, IT process automation is the key to industrializing your operations to improve productivity and reduce costs." (Garbani & O'Donnell, Market Overview: IT Process Automation, Q3 2011, 2011)

A given IT organization has a huge variety of tasks that are done repetitively. These tasks vary from something as simple as running a single command to update a single file on a single operating system every day, all the way to a process that involves several hundred independent steps across several thousand unique systems or groups (collections) of systems. The approach to automating both types of process share similarities, but the scope and complexity will ultimately drive the direction and level of effort the automation takes to develop completely. The Service Provider is our target stakeholder for which these kinds of automation tasks will be developed on and therefore the systems where an automation will occur are likely to span across many thousand distinct systems or system groups.

The following data represents some recent key statistical information about the state of IT Process Automation. The following surveys were conducted by Gartner Research over a 4 year period to "provide an understanding of the role process automation plays in IT operations, showing trends for automation tools investment, use cases, planned investment needs, vendor and product choices." (Williams, 2010)

Some of the key findings are shown below (Williams, 2010):

- *Fueled by difficult market conditions, the reduction in human interaction and IT operations costs are the highest-rated reasons for implementing process automation tools.*
- *IT operations process automation maturity has improved, with IT organizations moving from a basic state (job scheduling, basic task automation) to a moderate state (multiple nonintegrated workflows in IT operations management silos).*
- *IT operations process automation (Runbook Automation, RBA) tools have become mainstream, with market adoption influenced by large vendors using them to integrate, enhance and augment their configuration provisioning, fault and problem management products.*

Reviewing the results of the survey one can see a few trends that are worth noting. The first is that reducing human intervention, reducing cost, and reducing risk are the key motivating factors for an IT organization to implement a ITPA strategy, and that automation of Provisioning, configuration management, and fault or problem management were the top priorities that organizations felt deserved the highest focus. The following table extracted from the Gartner report summarize the findings to the question "What are (or would be) your primary reasons for adopting or considering IT Process automation?"

Top Three "First Priorities"		Top Three "Second Priorities"		Top Three "Third Priorities"	
Reduce human intervention	40%	Risk mitigation	29%	Reduce complexity	25%
Reduce IT operations cost	19%	Reduce IT operations costs	27%	Reduce human intervention	21%
Reduce complexity	15%	Reduce human intervention	20%	Support a virtual/cloud init.	18%

Table 3 - Top Three IT Automation Priorities Survey (Williams, 2010)

The full results of this question are shown below:

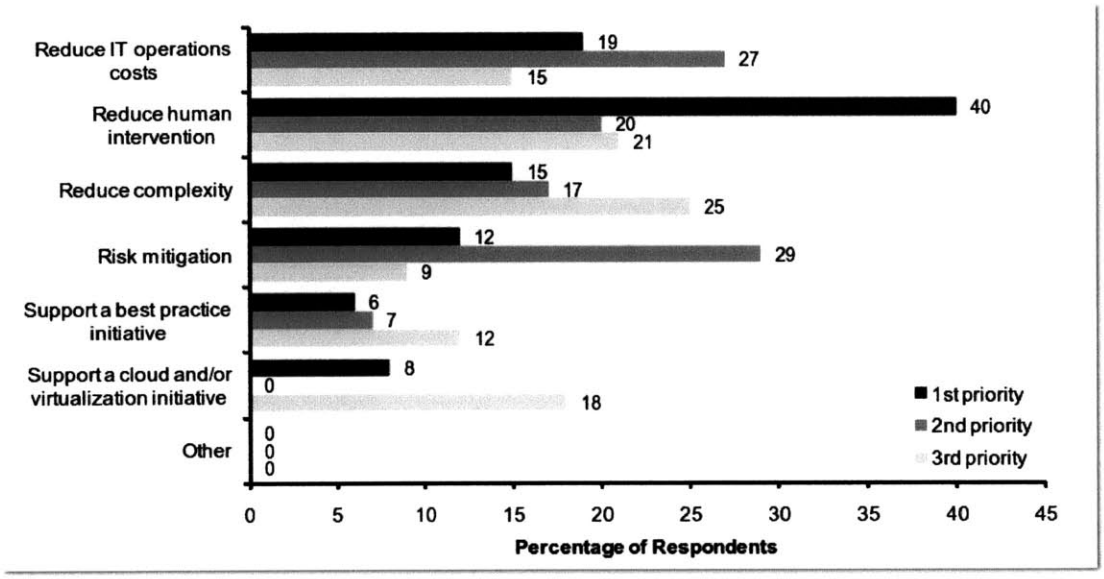


Figure 7 – Survey - What are (or would be) your primary reasons for adopting or considering IT process automation? (Enter top three in priority order)

Several other key findings from the survey are shown below to the following questions.

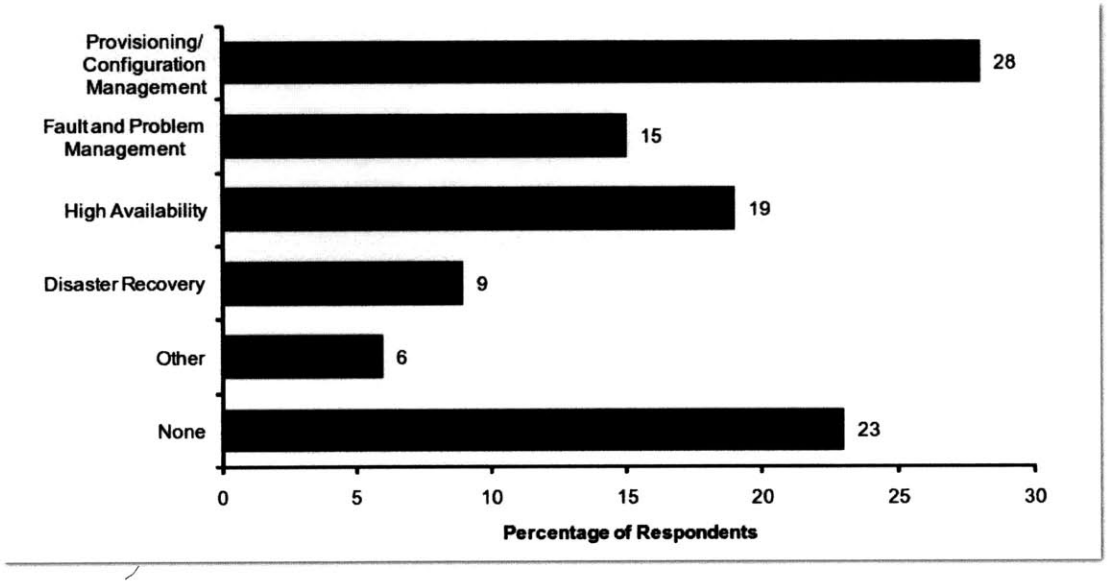


Figure 8 – Survey - Which disciplines have you already automated (or are in the process of automating) (Enter all that apply)

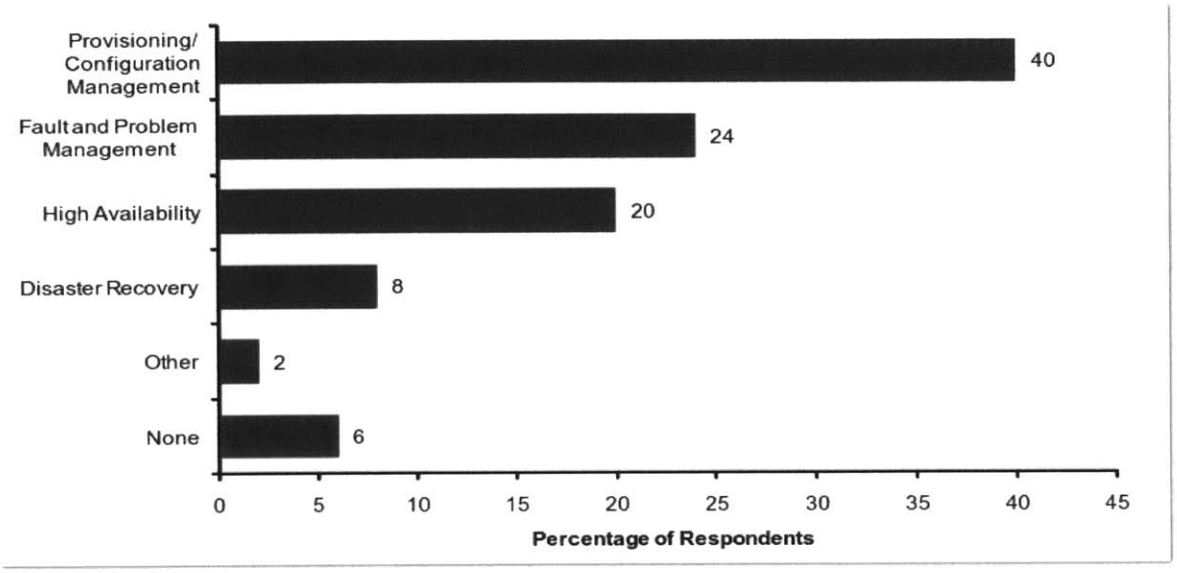


Figure 9 – Survey - What is your highest-priority focus for IT operations process automation for the next 12 to 18 months? (Select 1)

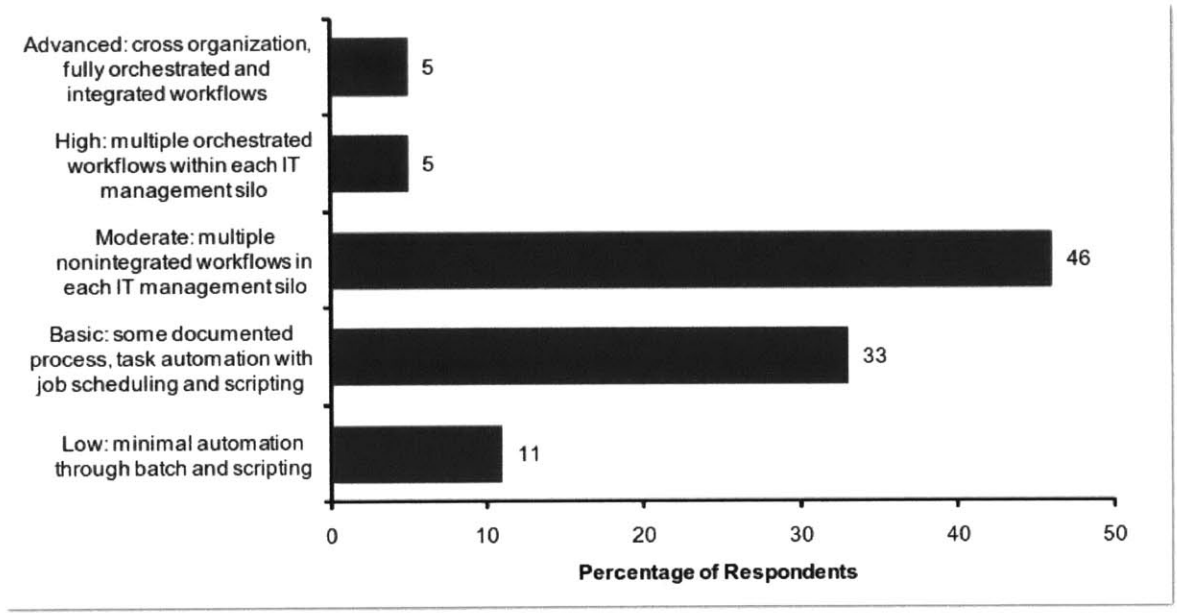


Figure 10 – Survey - How would you assess you IT operations process level? (Select 1)

The next section begins to investigate some key approaches to process automation and workflows to help address some of the needs illustrated above.

Approaches to Automation

IT Process Automation is not a plugin piece of software that is installed and begins to function magically for your organization. ITPA is an evolutionary and likely involves a continuous process of improvement. The operational capability and the process maturity of an IT organization will be reflected in the value provided by ITPA. For example, if an organization is slapdash and slipshod with its approach to creating and implementing their ITPA the result will be equally disordered. None the less before we can investigate how to automate IT processes we must first understand the process that is going to be automated. The greater an automation architect understands given processes and workflow, how they work, and what can go wrong, the greater the result will be. This seems obvious but is often missed when architecting an organization's IT Process Automation solution.

In this section we start by inspecting process automation at a very straightforward level by digging into a simple two-process workflow. The processes will run on a single operating system, (The OS type is immaterial) on a single server, (Again the type here is also immaterial to the discussion) and will perform a very simple task of a character string search and replace as shown below. These processes could really do anything, but for the purpose of our investigation they will have the following properties:

1.) Each process in the workflow below modifies a specific file on the operating system and replaces all strings "Bart" with the string "Lisa" by executing Process "A". Process "B" will similarly change all strings "Homer" with the string "Marge" in Process "B"

    For example: If the target file for Process "A" is `/some/directory/myTestFIle.txt` and has the content "Bart runs with Lisa to school" then the following would result when Process "A" is executed:

Content before Process "A" is executed	**Content after Process "A" is executed**
Bart runs with Lisa to school.	*Lisa* runs with Lisa to school

2.) Process A and B complete near instantaneously.

3.) Process A and B are Idempotent (This is implied by the nature of the Process string replacement, however Idempotence is described in more detail below) In general this property means that our processes will do what they need to do only if they need to do it otherwise they will not. This allows them the ability to run repeatedly, safely. For example Process A can run a thousand times but it's only the first time that it will modify and subsequently write or "commit" any changes to the file. Every

other time the string to be replaced will not be found, and the process will exit gracefully without making any changes. Idempotence is defined below (Anonymous, 2013):

- **A unary operation (or function) is idempotent if**, whenever it is applied twice to any value, it gives the same result as if it were applied once; i.e., **f(f(x)) ≡ f(x).**

  For example: the absolute value: **abs(abs(x)) ≡ abs(x)**.
- **A binary operation is idempotent if**, whenever it is applied to two equal values, it gives that value as the result.

  For example: the operation giving the maximum value of two values is idempotent: **max (x, x) ≡ x.**
- Given a binary operation, an idempotent element (or simply an idempotent) for the operation is a value for which the operation, when given that value for both of its operands, gives the value as the result.

  For example: the number 1 is an idempotent of multiplication: **1 × 1 = 1**.

The preceding properties are by no means required by any process in an automated workflow, but are used in this example to help understand the level of detail that must be considered even with the simplest processes. Some example code that could represent the innards of these processes are shown below:

```
//PROCESS A
IF TARGET FILE EXISTS
THEN
        IF FILE CONTAINS STRING "Bart"
        THEN
                REPLACE STRING "Bart" WITH STRING "Lisa"
        ELSE
                DO NOTHING AND CONTINUE
        FIN
ELSE
        EXIT WITH NULL POINTER ERROR
FIN
EXIT WITH SUCCESS STATUS
```

NOTE: It should be clearly understood that there are several other error conditions that can exist beyond a "file not found condition" however for the purposes of simplicity and illustration it is constructed this way to show a single failure condition for these particular processes. Process B would look nearly identical but use different string replacements.

Now that we understand a bit more about the processes we are automating we can begin to look at putting them together in a workflow that will execute on a single server and operating system.

We start by "daisy chaining" these two distinct processes on a single operating system (virtual or physical) and try to understand what is involved in design this very simple automation workflow. Review below:

Two Process - Single System Workflow

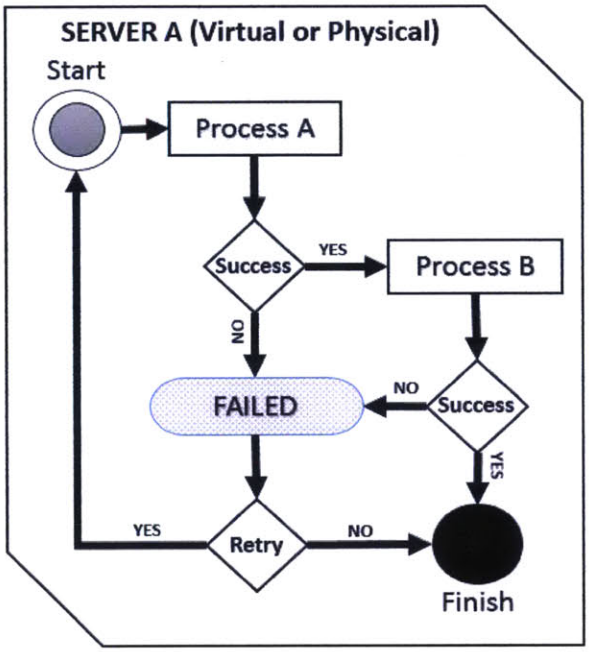


Figure 11 - Two Process - Single System Workflow

Conceptually the act of linking two simple IT processes together seems simple. This fact can trick the process automation designer or manager into thinking that creating a process workflow for ITPA is also simple. In the examples here however we are starting with 2 processes that are simple to build our understanding at an atomic level and build out. Later we will expand into process workflows that involve several dozen processes that span multiple systems within a "group" of systems, and then show how to automate and track those process across hundreds if not hundreds of thousands of "groups" of systems.

With this fact its fair that we break apart this simplified automation case for the purpose of pointing out some of the subtleties that the automation architect will need to grapple with in the workflow. The pseudo code below is one way to represent algorithm of the workflow diagram shown above. This pseudo code will ultimately need to run on the system where the IT Process workflow is executed. It will also need to report the status of the workflow that has been executed to the controller of the workflow. For example:

Process A and B workflow

```
START ITPA_PROCESS_1
IF PROCESS A SUCCEEDS
THEN
       IF PROCESS B SUCCEEDS
```

```
        THEN
                FINISH EXIT SUCCESS CODE
        ELSE
                IF RETRY ON FAILURE
                THEN
                        RESTART ITPA_PROCESS_1
                ELSE
                        FINISH EXIT FAILURE CODE FAIL STEP 2
                FIN
        FIN
ELSE

        IF RETRY ON FAILURE
        THEN
                RESTART ITPA_PROCESS_1
        ELSE
                FINISH EXIT FAILURE CODE FAIL STEP 1
        FIN
FIN
FINISH EXIT FAILURE TIMEOUT OR MAX RETRIES
```

With this example above one can see that even in this uncomplicated case there are several paths this process automation workflow could take to finish. (Successfully or with failure) For example:

- Finish both processes successfully and exit successfully.
- Finish neither of the two process successfully and exit with failure.
- Finish one processes successfully while the other fails and exit with failure.
- Finish with a timeout or iteration failure due to max retries.

Some of the many questions that need to be considered in this situation by the automation architect are shown below:

- What constitutes the success or failure of a given process?
    - Can a process partially fail or succeed?
    - How does the automation respond in the event of a partial failure?
- How many times can a process retry before it gives up?
- What mechanism will the workflow use to report back its result to the operator or controller?
- Are the processes retry-able or idempotent?
- Can the processes be "undone"?
- Is the whole workflow retry-able or idempotent? (This assumes Idempotence for all Processes)

Each of these questions above are not just considerations to be theorized about when designing the automation for this workflow, they must be spelled out in exact detail within the automation code in order to have the automation perform as it was intended by the automation architect.

Let's now take some time to view this same simple two process workflow and split it across two systems and observe how the workflow diagram has changed:

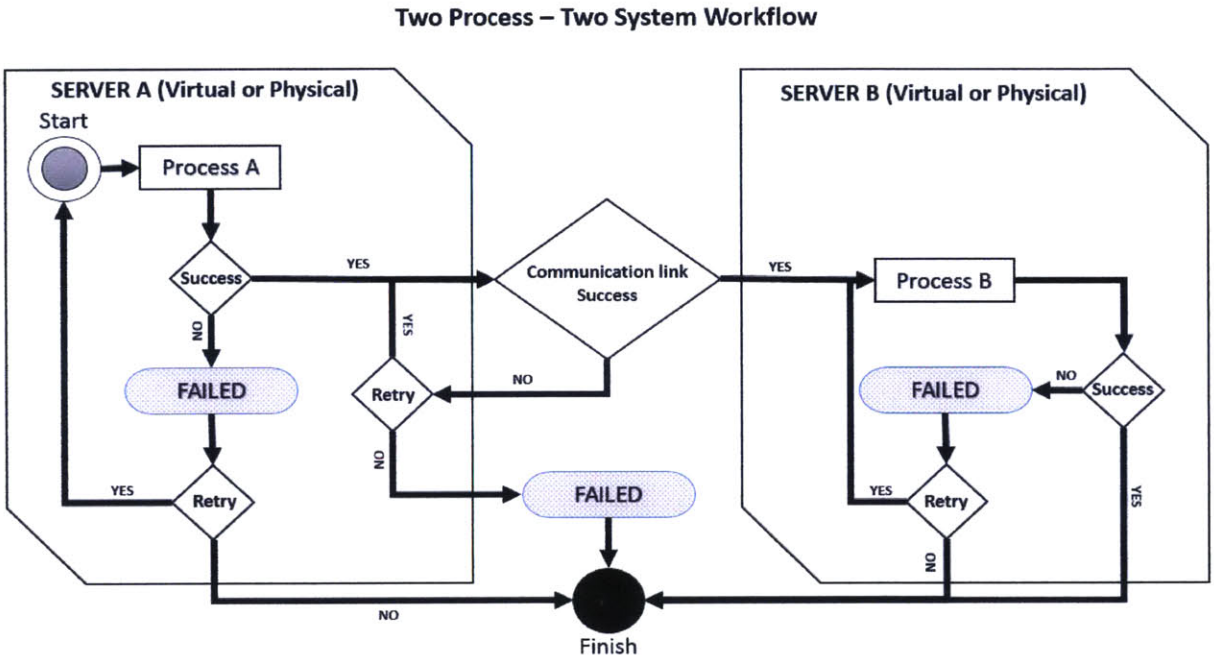


Figure 12 - Two Process - Two System Workflow

In the circumstance above the number of processes that are involved in the workflow have not changed from the previous example and for the most part the workflow logic has remained the same outside the introduction of a "communication link" (Interface) between the two systems. This communication link is an extremely important element in this new workflow however and creates several problems for the process architect to solve. The communication link has created several ambiguities in the workflow process for the operator, the support staff of the workflow, and possibly to the upstream controller (human or machine). For example the following questions now need to be asked:

- What are the properties of the communication link?
- How will the processes communicate through the link?
- Where will the status of the workflow exist at any given point? (EX Server A, B, or somewhere else?)
- How is process control passed?
- If the processes were long running procedures how would one be able to tell what the process status is at any given point in time.

These are obviously only scratching the surface of the complexities of passing control "intra" systems where the systems exist as part of a group. Grouping systems together particularly in complex services is somewhat unavoidable at present, and therefore creating automated workflows that spans a great many systems within a group is common.

Below we will create a final example workflow that performs the following actions

Three System Seven Step ITPA Workflow Example:

- **Process A** – Stop a webserver on **SERVER A**
- **Process B** – Stop an application server on **SERVER B**
- **Process C** – Stop a database server on **SERVER C**
- **Process D** – Update a configuration file on **SERVER A, B, and C**
- **Process E** – Start a database server on **SERVER C**
- **Process F** – Start an application server on **SERVER B**
- **Process G** – Start a webserver on **SERVER A**

These steps are each fairly common processes and somewhat straightforward for any operations team or system administrator. They should be able to perform these tasks at the command line interface (CLI) of any given system with ease. However when defining the ITPA workflow for these seven processes across three machines, the complexity magnifies. The models used previously will become far too complex and cumbersome to continue to implement in the same manner. For example imagine the complexities involved in inter system communication links and communications that must occur between the systems to keep tally on what process is doing what and when. At this point we need to limitedly introduce the concepts of an **External Orchestrator** and an **External Orchestrator Agent**, though they will be discussed in further detail in the section below. The External Orchestrator and the External Orchestrator Agent are critical elements of our ITPA process controller. These elements will need to communicate via some method across systems order to perform the following two main functions.

1.) Initiate (or execute) a process on a given Server and,
2.) Report and pass status messages from the External Orchestrator Agent to the External Orchestrator at any given moment following the initiation (or execution) of a given process.

The External Orchestrator (in this context) can either be human or machine and must be able to intercept and interpret the messages from the External Orchestrator Agents. It must also be able to understand and

control the flow of events from a top level orchestration perspective. The logical flow of events in the seven process – three system workflow described above is shown below in diagram form.

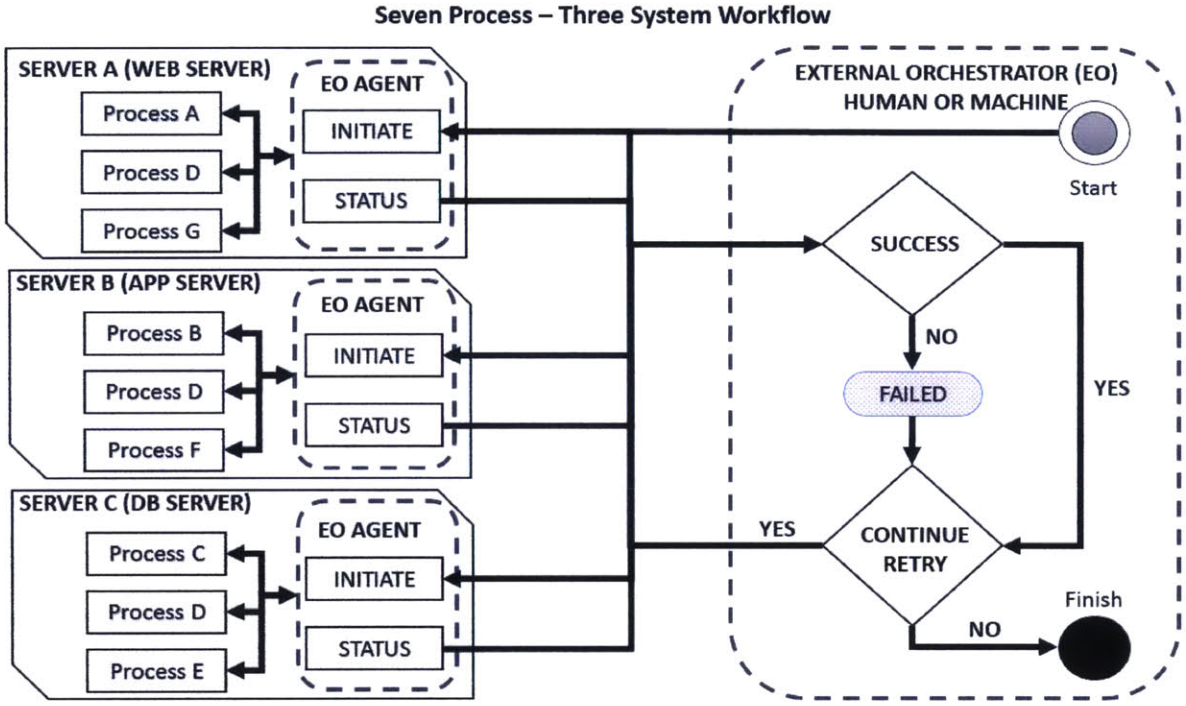


Figure 13 - Seven Process - Three System ITPA Workflow – The External Orchestrator

From this diagram it is somewhat easy to see that the number of ways that even this simple process flow can finish and exit far exceeds the number of ways that the Two Process – Two System workflow can finish and exit. The design of the automation at an architectural level must be considered very carefully at this point to in order to remove as much complexity and ambiguity from the process workflow as possible. If not, the operators of the system's automation scheme may find themselves unnecessarily troubleshooting a variety of issues as changes are made to the ITPA workflows.

At this point it should be understandable that the complexity of automating of even a small set of standard IT processes can become very complicated very quickly. The processes referenced in this section, though highly simplified, are enough to provide (at least at a high-level) a view of the types of automation that will be coordinated via an ITPA workflow. In actuality, IT processes particularly for a Service Providers will be significantly more complex, and will be managed across many thousands of discrete "groups" of systems and often times in parallel.

Describing an approach to the automation and parallel execution of IT processes across many groups of heterogeneous system is one of the primary goals of this paper and we begin this discussion in the following sections.

Orchestration and Automation Products

The market if full of pre-packaged ITPA and Run book Automation "solutions." These solutions can be deployed in order to address any given set of process automation needs. Several of the vendors offer "canned" or templatized workflows for organizing a specific workflow or process. There are also a great variety of modules that can be used to tie into an existing monitoring solution or dashboard common to an IT organization. Though it is helpful to understand the commercial landscape the main focus of this paper is to understand a particular methodology of IT Process Automation that is not bound by any particular vendor or an ITPA or Runbook product. None the less, the following is a brief overview of the ever increasing landscape of ITPA and Runbook commercial vendors. There is also a growing variety of open-source solutions as well. In a recent Gartner report (Colville, IT Process Automation: Run Book Automation Tools Mature to Broader Use, 2011)the following are some of the key players in the ITPA space each offering approaches to the automation needs of any given IT Providers organization

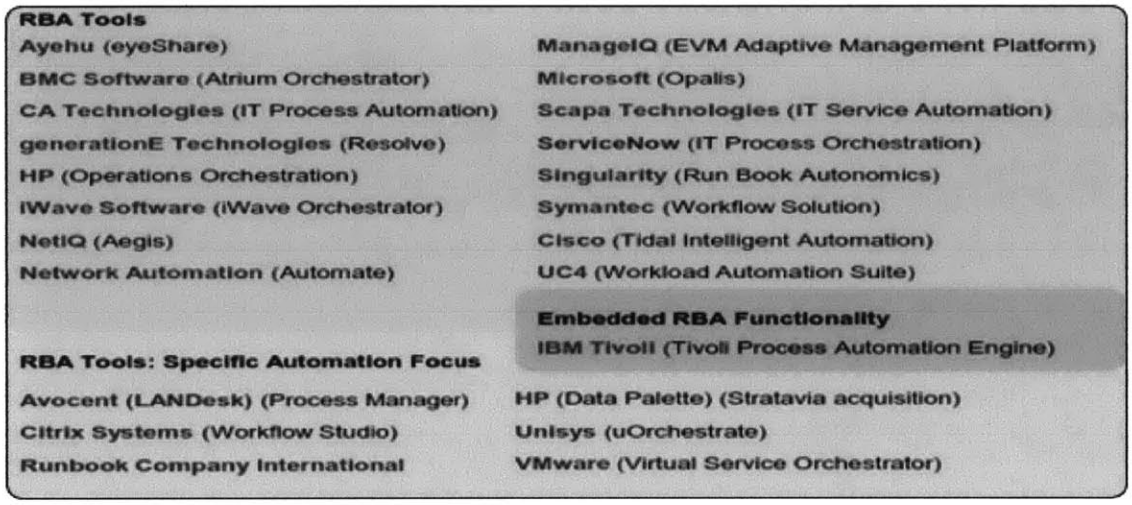


Figure 14 - ITPA - RBA Gartner Vendor Landscape

In the open-source landscape is equally filled with canned solutions and tools to help an enterprise customer automate workflows and processes. Some of the open source vendors include PuppetLabs, OpsCode, Salt Stack, and several others. The solutions offered by open source vendors offer the ability to

create a high degree of customization for a particular ITPA solution. This advantage is due in part to the ability to manipulate the code base to accommodate special needs of a particular organization.

Though the focus of this paper is on the creation and use of process automation specific to large scale IT Service Providers, the findings and recommendations by Gartner (Colville, IT Process Automation: Run Book Automation Tools Mature to Broader Use, 2011) for all IT vendors (with relation to ITPA and RBA) are still very much applicable to our case. The recommendations will apply to any variety of ITPA product open source or otherwise; specifically:

Gartner - Key Findings

- Use cases for RBA tools have expanded beyond the initial task automation.
- While ITPA tools have some out-of-the-box workflows, they still require customization and scripting to address all aspects of your unique process requirements.
- Because there are no ITPA standards and it is difficult to bridge automation from one system to another, organizations will likely end up with multiple tools to address different process problems.
- New use cases for ITPA continue to emerge for specific needs around release automation (e.g., DevOps) and cloud projects.

Gartner - Recommendations

- Know your process. These tools offer some out-of-the-box workflows, but none will completely match your processes. You must document your process workflows before selecting an ITPA tool.
- ITPA tools require specific, assigned responsibility and dedicated expertise.
- Understand your tool integration points. Most processes include touch points or triggers to other management tools. You must understand what management tools are in scope for the process being addressed before selecting an ITPA tool.
- Start your ITPA projects narrowly. No ITPA tool will automate all IT processes, thus obviating the need for people or domain-specific tools.

Customization and deep understanding of the processes being automated are essential to the success of any given ITPA/RBA strategy or rollout. This is particularly true for IT Service Providers that will need to produce workflows and automations that in all likelihood will be replicated across many thousands of distinct services of varying complexity.

For the most part the pre-baked orchestration paradigms and templates will offer many ideas on how an organization can build their process automation, but these templates may not necessarily fit an organization's needs. This paper's messaging concepts (available through a variety of different products) will help to formulate a framework and architecture that can support a highly customizable and scalable ITPA and Runbook solution for many types of IT Service providers. So the solutions offered here are not specific to any particular automation workflow, but rather an execution framework that will enable any number of process automation workflows in a large variety of circumstance. Before diving into the specifics of this framework we should review what we want our framework to accomplish. This is discussed below.

To review, there were two main dimensions discussed in the first chapter that dealt with complexity that need to be addressed again here in the context of the ITPA framework; vertical and horizontal complexity. Each of the products highlighted previously can handle one or both of these dimensions in one way or another. But, in the following sections, we will discuss some of the key types of product-agnostic technologies that can be combined in particular ways that will provide the building blocks to create an ITPA solution that will scale in both directions.

In order to understand how one might achieve these ends, we move back to the concept of an external "orchestrator." Some of the key goals of a core orchestration system are shown below:

1.) The orchestrator must be able to execute any arbitrary operation, on any given system (or group of systems) within its domain of influence.
2.) The orchestrator must be able to obtain the status (or result) of the operation.
3.) It must be able to execute operations both sequentially and in parallel to any system or group of systems within its domain of influence. (NOTE: This is a distinct need from #1.)
4.) The orchestrator must provide a mechanism that enables the operator or workflow architect execute both ad-hoc (from the command line interface CLI) and programmatic control logic (automated workflow) both vertically and horizontally.

There are several methods that can be used to achieve these ends and one could use any variety of "client-server" based remote procedure (RPC) calls to communicate their execution payload to the intended systems. Loops using these coupled technologies such as "ssh (secure shell)", "rpc(remote procedure call)", or some other method are fairly straightforward particularly when the types of systems involved are homogeneous and the processes being executed are somewhat simple and quickly return a response.

The tightly coupled client-server messaging model puts a lot of responsibility on the external orchestrator, particularly as it relates to maintaining a central meta-data repository and maintaining sessions between the orchestrator and the orchestrated systems. Additionally, designing the algorithms, the requisite metadata (and workflow) repositories, as well as determining who, what, when, where, why and how things get executed is not only complicated to design and implement, but also very difficult to operate in an efficient way. However, an alternative approach to using a tightly coupled "client-server" based messaging scheme (via ssh/rpc/etc), the recommendation is to use a decoupled messaging technology. The type of decoupled messaging technology we will use to build our ITPA orchestrator is through **Publish-Subscribe**. Publish-Subscribe technology allows the external orchestrator to become a decoupled messaging bus. The infrastructure implementation of a messaging bus can be achieved by using message queuing (MQ) software that supports publish-subscribe as a messaging pattern. The is a large list of message queue commercial off the shelf products and a few are IBM WebSphere MQ, JMS, RabbitMQ, ActiveMQ, and Microsoft Message Queuing. They are equivalent technologies from the perspective of developing our ITPA platform.

Below are some ideas of architectures for how an ITPA orchestration framework can be implemented using a publish-subscribe and broadcast paradigm as its core messaging mechanism.

Building a Messaging Framework through Publish-Subscribe and Broadcast Paradigms

To simplify the design process of our external orchestrator we will investigate using publish-subscribe as our messaging broadcast paradigm. The orchestrator will act as a core "messaging" bus and hub in addition to providing the execution logic and execution interfaces for our IPTA architecture. The messaging patterns of publish-subscribe make it an ideal candidate for our external orchestrator because of the loose coupling between the publisher and the subscriber as well as its ability to scale. The reasons for this choice will become more evident as we build our ITPA external orchestrator and describe its design and use.

Publish-subscribe is not a new technology for intra-application messaging schemes. In fact, it was first described in a paper titled "*Exploiting virtual synchrony in distributed systems*" (Birman & Joseph, 1987). Typically publish subscribe technologies have used as part of an application's messaging architecture and generally publishers and subscribers interface within a particular application's context. Very often that application would be housed within a single operating system or a tightly grouped set of operating systems. Until recently however, publish-subscribe has not been generally thought of as a tool for managing the operating systems themselves or the execution workflows within operating system's

context. This is starting to change perhaps in response to the sheer number of operating systems being built as part of a particular service architecture. IT Service Providers system infrastructures can span into the millions of virtual and physical servers or Operating Systems. The somewhat recent advances virtualization technologies have changed the way Operating Systems are used within a Service Provider's datacenter. The 1:1 mapping between the operating system and single piece of hardware is no longer a good model of modern computing. A single piece of hardware will generally now house many operating systems, each with their own processor, memory, and file system management mechanisms through the OS kernel. Moreover operating systems these days are moving more into the realm of being reusable "modules" of complex applications, rather than the core infrastructure that houses and runs the whole application. This is perhaps due to the fact that operating systems can now be created in a way that blurs the lines between application installations and operating system installations. It should also not be assumed that an application will run within the confines of a single operating system. Perhaps the following simple illustration shows this.

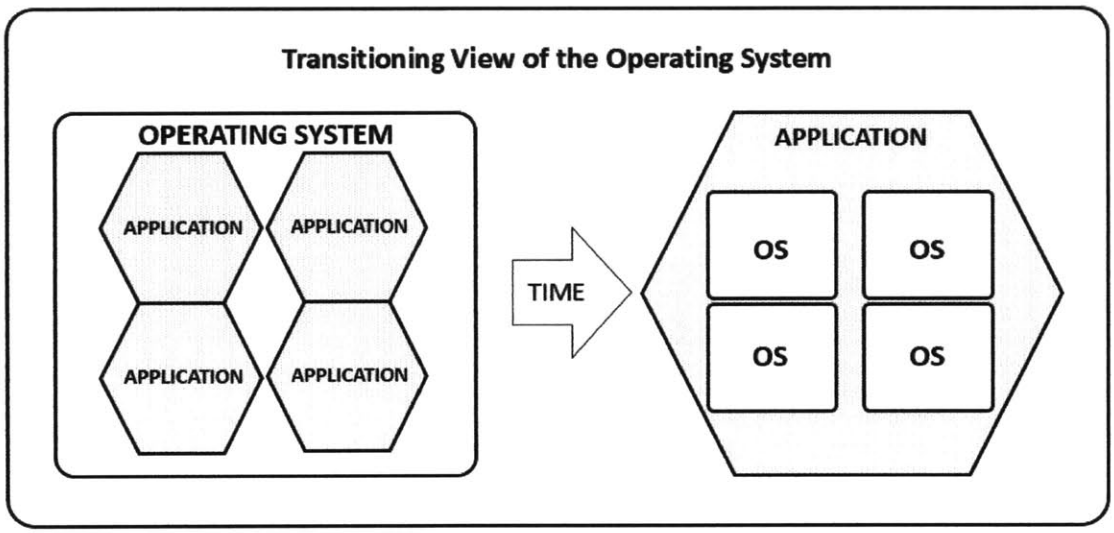


Figure 15 - Transitioning View of the Operating System

This more recent and common design pattern has created a need to rethink the way we IT organizations communicate with their systems, groups of systems, and how they manage the applications run on them. An orchestrator must have the ability to communicate fluently with both the operating system and the applications. It is generally easier to communicate with an application through the operating system then to speak to the operating system through the application. Therefore the most efficient path to "get to both" is through the operating system. Additionally a vast majority of IT processes will be kicked off from

the operating system rather than through a specific application mechanism, particularly for infrastructure operations that involve the state of both.

Publish-Subscribe works in a general sense by providing a shared location where a particular message can be placed and subsequently read by any number of clients. That location is called a "**topic**" and must be accessible by both the orchestrator and the nodes (or operating systems) that it orchestrates. Most times that topic exists as a queue on a message "**Broker**", such as those provided as part of a message queuing service (MQ). "**Publishers**" publish messages to the topic and once the message is in place any number of "**Subscribers**" can read the contents of the message, in part or whole. The architecture is loosely coupled given that the publisher does not need to have knowledge of the subscriber and the subscriber does not need to have knowledge of the publisher. The following diagram illustrates this generic message flow:

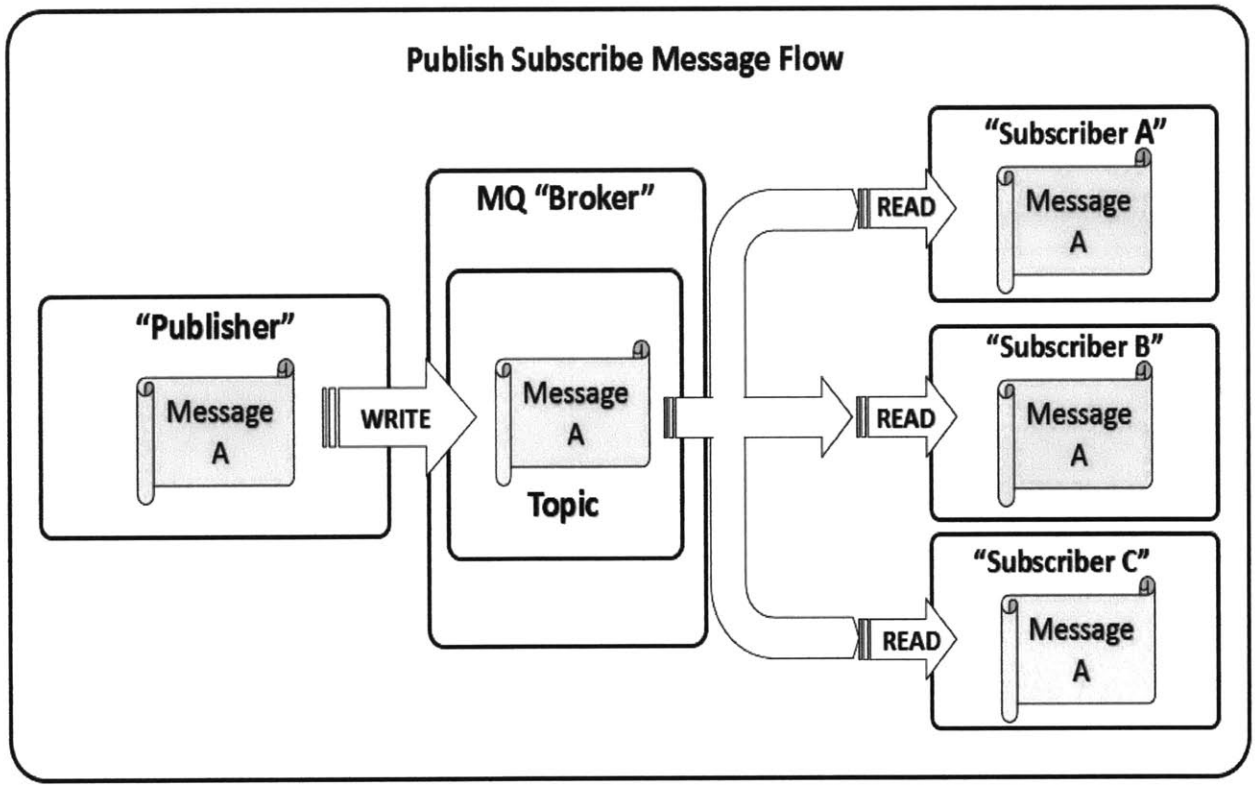


Figure 16 - Publish-Subscribe Message Flow Example

Of course this is an oversimplification for how publish-subscribe works, but one can begin to see that only one message is written to the topic queue and is then read by multiple recipients. This moves the responsibility of message "delivery" from the publisher to the subscriber and in essence distributes the messaging load. There of course can still be contention at the broker for very large data centers so

consideration should be given to the compute power of the broker in relation to the size of the systems fleet.

The subscribers are for the most part responsible for which messages they read and respond to by filtering their messages. Filtering can occur at the topic level, or based upon a particular "addressing attributes" embedded in the message. (This is described more in the section below) The beauty of this design pattern is that it will enable you to specify from the publisher's standpoint which system or groups of systems you would like to respond to or act upon your message by tweaking the inclusive (or exclusive) attributes in the message. This is a very important concept to grasp from an ITPA Operational perspective and so the point is illustrated in more detail below. It is also important to understand that for each message received and acted upon by a subscriber, the publisher will in turn receive a reply message from the subscriber indicating that it has received and is responding to the message. The specifics of this message-receipt flow will be dependent on the implementing software, however several products that handle this type of messaging paradigm use the "streaming text oriented messaging protocol" (STOMP) specification to facilitate this message-receipt coupling in order to abstract the messaging MQ subsystem. This specification works in a similar way to HTTP and includes generic functions to "CONNECT, SEND, SUBSCRIBE, UNSUBSCRIBE, BEGIN, COMMIT, ABORT, ACK, NACK, and DISCONNECT" (Creative Commons Attribution v3.0 , 2014)For the purposes of the following discussion we can assume that regardless of the implementation, any message "sent" via the publisher will in turn receive a "response" acknowledgement from an individual subscriber. It is within that acknowledgement envelope that the messages from the individual subscribers will be embedded.

Let's begin by defining our small fleet of systems. The fleet consists of 9 groups of systems each with 8 servers associated with each group. Each group of systems runs a single logical application. So one application belongs to one group. In total our very small fleet consists of 72 servers, but could just as soon be 720, or 72,000. Each server in the fleet is a subscriber to the "execution" topic that exists on the central message broker. They will listen on that topic for messages targeted at them. The publisher for this first action that we will define in MESSAGE A will create the following message content and deliver it to the intended "execution" topic:

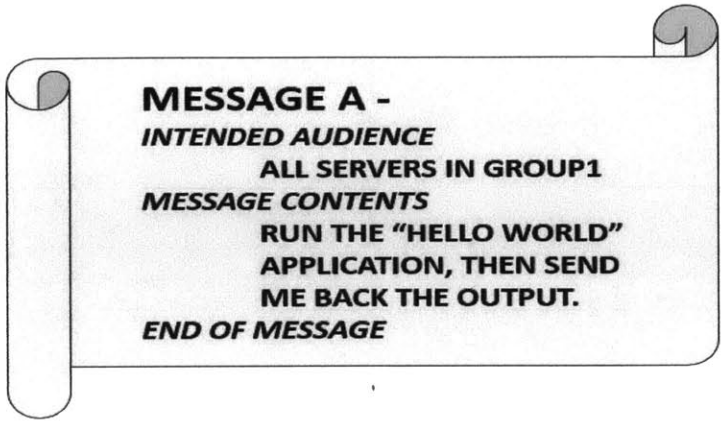


Figure 17 - "Pseudo" Message

The expectation at this point is that once the message is put on the topic, all subscribers will be able to view the message. However, only the subscribers matching the "*Group 1*" attribute will "RUN" and "SEND OUTPUT" (Which would be whatever the output of the "HELLO WORLD" application is.) It should be noted that there are a variety of customizable products on the market that are generic enough enable this kind of message flow between the publishers and the subscribers within in a fleet of systems, namely mcollective and Salt Stack. (Both open source products) The implementation of how a message is handled is highly customizable and therefore is up to the imagination of the automation architect and workflow designers. Regardless, the ability to pass these arbitrary messages to targeted systems is one of the key building blocks to this papers proposed ITPA workflow automation scheme.

Addressing the Systems in Fleet with Attribute-Value Pairs

Targeting systems is another key feature of the ITPA system being developed. In other words the ITPA system must be able to tell specific systems or groups of systems what to do at any given time. "Addressing targets" was briefly discussed in the previous publish-subscribe section, but will be illuminated in a bit more detail here for its importance to the overall ITPA architecture.

Again, Publish-Subscribe was chosen for two main reasons; the decoupled nature of publish-subscribe messaging pattern and the ability for publish-subscribe to scale. However, if the publisher is decoupled (or has no real knowledge of its subscribers), then how will it know where to address its messages. To solve this publish-subscribe requires somewhat counterintuitive approach to address or target messages to specific systems within the fleet.

The traditional approach of messaging in a coupled client-server model the message is delivered directly to a known address, or addresses. A separate message is delivered directly by the server to each client. For example the server knows precisely to whom it will deliver the message. If a coupled client-server model were chosen for a large fleet ITPA system it will require that the server side have a large meta-data repository with information about each system that existed in the fleet. For example: If I had 200 servers that were of type A and another 200 servers of type B, I would register those systems in my meta-data repository as having either type A or type B. This is a clunky system for several reasons. Firstly what will happen when 50 of those servers of type B are migrated to be of type A. At that point you must reconcile your repository to reflect that fact. It seems simple enough in this straightforward situation, but imagine if the fleet consisted of many thousands of systems each with hundreds of shifting attributes at any given point in time. The nature of an IT Service Provider's system fleet is one of constant flux with systems constantly being modified and moving in and out of the fleet. To try to manage and keep current in a meta-data repository that information, is at best error-prone process and most likely a constant exercise in frustration. This is why the publish-subscribe paradigm works well in very large constantly changing system fleets; the subscriber is its own system of record and therefore there is no need to create a centralized repository that holds all of the information about each system. In publish-subscribe each system is responsible for responding *only* to what is directed specifically to it, and it is the final arbiter.

A Rock Start Analogy for Attribute Based Responses in Publish-Subscribe:

A Rock Star is standing on a stage for a concert in Chicago, in front of 50,000 fans. The Rock Star is not aware of who is in audience but he can still query the audience to gather responses based on the attributes of the individual members of the audience. Here's how, the Rock Star shouts "Anybody here from Chicago?" Clearly the fans that are from Chicago will cheer loudly and will undoubtedly represent a fairly large portion of the audience. Then the Rock Star shouts "Anybody here to Party?" Perhaps an even a bigger group of fans will begin to cheer. However to extend this analogy, perhaps somewhat absurdly, what if that same Rock Star shouted "Anybody here over the age of 30 with Brown hair?" Clearly the participation would be slightly muted but still would undoubtedly include a fairly large collection of people. However, if the Rock Star then should "Anybody here exactly 30 with Brown hair whose name begins with the letter S and is 6 feet tall." At this point in time there will certainly be some shouts coming from the audience in random places but most likely a very small minority of the total audience. Finally our Rock Star shouts "Anybody here whose name is Sam Sample, who lives on 1551 Sycamore Lane and drives a Toyota corolla with Illinois license plate 34X992? Your lights are on..." Now, either Sam is there or he is

not, but if he is there he will likely give a muffled "woot" from somewhere in the audience and then go turn off his lights. This may seem absurd in the context of a concert but this is exactly the way our ITPA system will query the systems fleet to determine who will response to what. These are attributes that only the individual system Subscriber (or "A fan") knows about. The Publisher ("The Rock Star") does not need to keep track of them, as it is the responsibility of the individual to reply or respond. Given that systems do not "lie" unless told to do so, one can assume that if a member of the fleet is there and asked to respond if it contains a given attribute, it will.

Key value pairs are a common attribute definition mechanism. In the example above each individual would have attributes like those shown below:

Table 4 - Possible Attributes Associated with a Rock Star's Fan

ATTRIBUTE	ATTRIBUTE VALUE				
NAME	JOE	SALLY	SAM	SUE	PAT
HEIGHT	5'6"	6'0"	6'3"	5'4"	5'10"
AGE	30	22	38	29	23

Each person is represented by one column and would have knowledge of only their "column's" attribute values. For example Sam would have the following information about himself: Nam=Joe, Height=5'6", Age=30. Sally however would have the following information about herself: Name=Sally, Height=6'0", Age=22. In the exact same way a system Subscriber will have a series of attributes associated with it that can be referenced within the message delivered to the Subscriber.

In our example of the 72 systems referenced in 9 group system fleet; each of the systems would have a file or repository containing the following information; Which SERVER LETTER they possess, and which SERVER GROUP they belong to. Each of these 72 files will contain enough information to make it uniquely identifiable and be enough to give it a group identity as well, for example.

Table 5 - Possible Attributes Associated with a System

ATTRIBUTE	ATTRIBUTE VALUE
SERVER LETTER	A
SERVER GROUP	1

These attributes can extend far beyond the two values we've included here to have other pertinent information about the system. Systems can be grouped by any number of attributes in isolation or in combination. The following shows some ideas of attributes that may be associated with a given system:

Table 6 – Several Other Possible Attributes Associated with a System

ATTRIBUTE	ATTRIBUTE VALUE
SERVER LETTER	B
SERVER GROUP	5
SERVER SYSTEM TYPE	SPARC
RAM	24GB
CPU	8 x Intel Quad @ 2.7GHz
DISK STORAGE	2.5TB
IP ADDRESS	192.168.100.20
MAC ADDRESS	00:0c:ff:ff:fc:e7
FULLY QUALIFIED DOMAIN NAME	serverb.somecooldomain.com
KERNEL VERSION	2.7.300.4
LAST MODIFICATION DATE	April 1, 2013
FUNCTION TYPE	Webserver
SYSTEM DESCRIPTION	Interchangeable webserver for somecooldomain fleet.

Now with the foundation of addressing the fleet of systems described, let's take a look at how one might use this to execute "arbitrary" commands to variable systems groups. Below are 6 example queries (or targeted broadcasts) that we will used to strategically call a particular execution to our systems. The execution will be something very simple like return a simple value like "uptime." The queries are as follows:

Table 7 - Forming Groups of Systems in the System Fleet

QUERY	DEFINED SYSTEM OR GROUP SET {}
A	{ANY SERVER IN GROUP 1}
B	{ANY SERVER IN ANY GROUP WITH SERVER LETTER A}
C	{ANY SERVER IN ANY GROUP 7 THROUGH 9}
D	{ANY SERVER IN GROUP 3 THROUGH 5 WITH A SERVER LETTER B THROUGH E }

E {ANY SERVER IN GROUP 3 THROUGH 5 WITH A SERVER LETTER G }

F {ANY SERVER IN ANY GROUP WITH ANY SERVER LETTER}

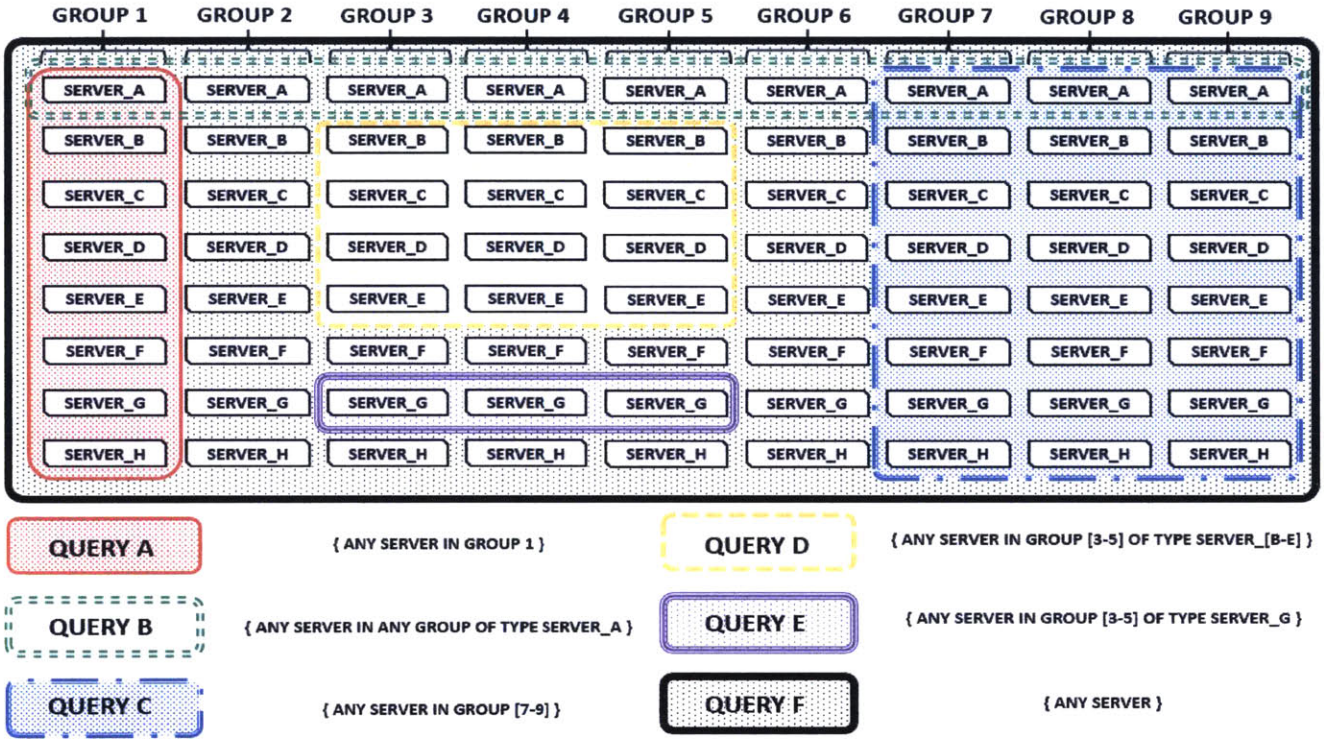


Figure 18 - Forming Groups through Attributes

At this point we have enough of a foundation to begin building a "Horizontal" automation approach to delivering execution or initialization messages to selective systems within the fleet. This type of approach will work very well where there are a huge number of systems within the fleet that are relatively simple and homogeneous in nature. However it can also be expanded to include highly complex groups of systems that are homogeneous in their group structure.

Automation via Horizontal "Group Based" Orchestration

Now that the ITPA system is able to send targeted messages to specific systems or groups of systems across the fleet, it is time to investigate some of the operational aspects of this new feature. Horizontal "Group Based Orchestration" is one way to address both horizontally complex environments (those that scale) and vertically complex systems by moving horizontally up and down the group or "stack of systems." There is an underlying assumption that the vertical complexity amongst each stack is similar. For example: If Server A from Group 1 is a webserver of a specific type then it is assumed that Server A in Groups 2-n

will also share a similar configuration. This is so that if you execute "Step 1" or "Step 2" horizontally across the fleet you are issuing identical "Steps" or commands against very similar systems. (See the section above on standardization and configuration management)

This type of execution motion, up and down the stack, is referred to as Horizontal Group Based Orchestration. Illustrated below:

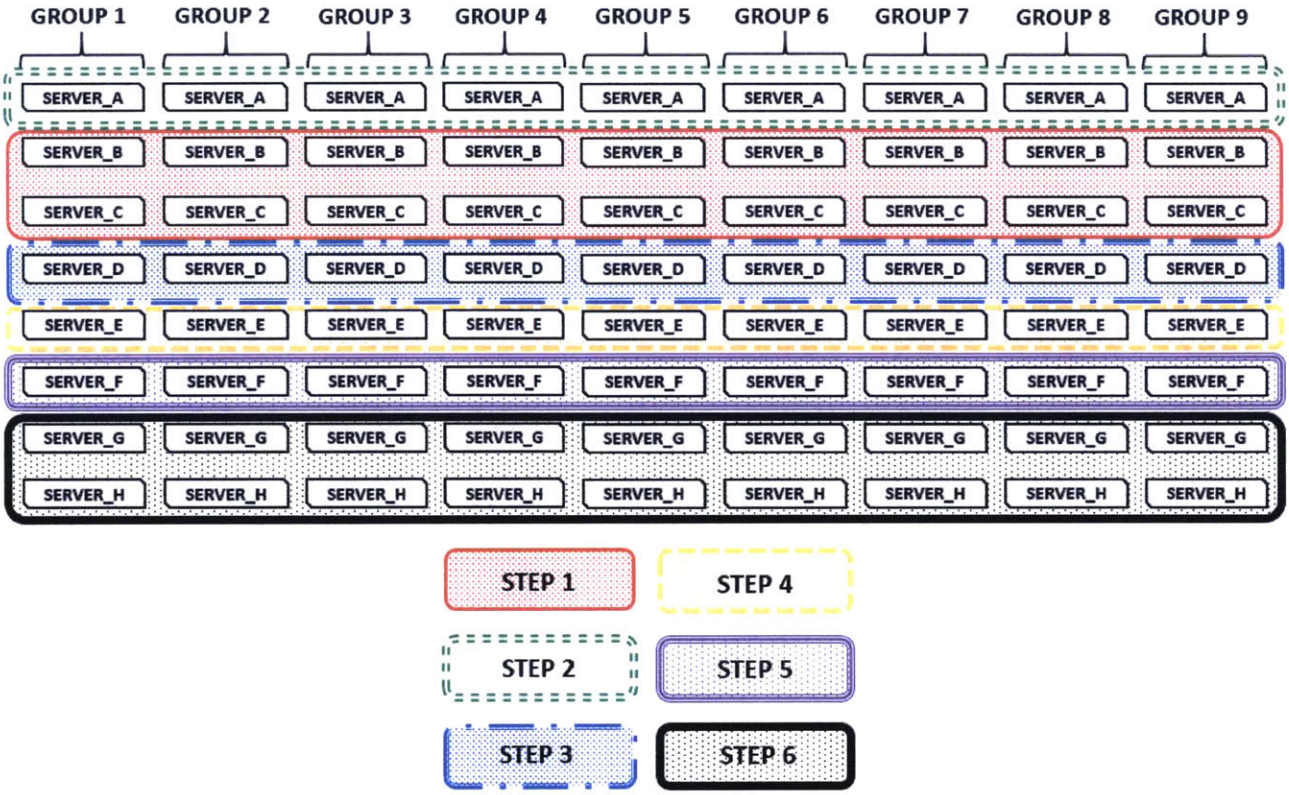


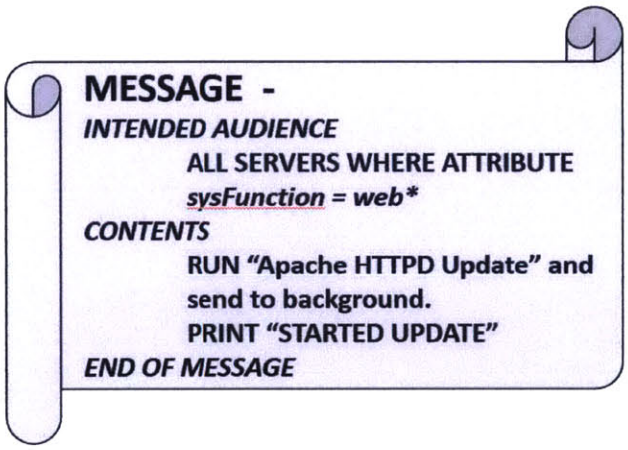
Figure 19 - Horizontal Group Based Orchestration

What can this capability offer to our ITPA automation and orchestration framework from an overall ITPA solution? The best way to discuss these capabilities will be through specific operational cases. Three are described below.

***Operational Case 1 – Application Change**: Update 10000 identical webserver systems with the latest Apache HTTPD Software.*

More of a recommendation than anything else, it's a good idea to create attributes that match the structure of your fleet. For example if the fleet is split by function then it is a good idea to have a function specific attribute that will enable the operators to tease out the intended systems from the fleet. In this

example will we have created an attribute on each system called "**sysFunction.**" sysFunction will have the value "webserver" on every system that performs the function webserver and has a similar configuration. In this case there are 10,000 of them in the fleet. Regular expressions (regex) are a common way of "querying" attributes via a variety of software packages that use publish-subscribe. As an example if there were only 3 types of systems in my fleet "webserver", "application server", and "database server" I could access all webservers by defining the message to look something like this:



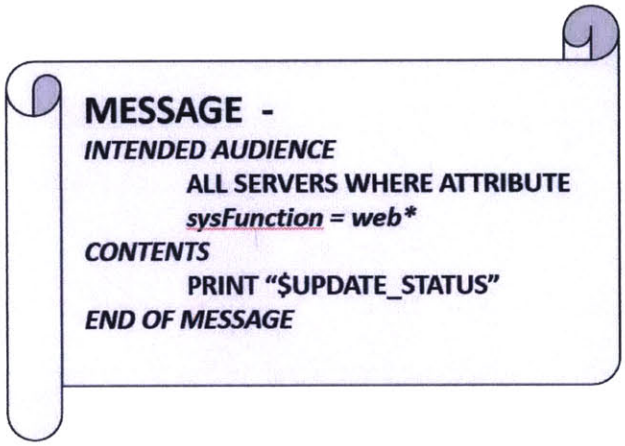
The Publisher will put this out on the "execution" topic and every subscriber to this topic will read this message. Because all of our webservers are subscribers, they will all read this message and realize that they are being asked to do 2 things;

1.) Run the Apache HTTPD Update process and background it*, and
2.) Return a message that says the system has started the update.

The reason why the running process is sent to the background is so that the system can immediately process the next step and return the message that the process has started almost instantly. What this will do is allow the publisher to receive a return response from all 10,000 systems almost instantly stating that the update has started. It is generally not a great idea to "wait" for a process to complete before returning a response to the Publisher given the variable nature of each system. For the moment all the Publisher (and the Publisher's Operator) needs to know is that the process was executed and not whether or not it succeeded in the execution. If the execution path is linear (e.g. first RUN (background) then PRINT) then knowledge of execution can be somewhat guaranteed by receiving the targeted response from all systems stating "STARTED UPDATE." Additionally any mechanism can be used to tally the responses back to the Publisher in order to ensure the "executed" number matches the intended number. A variety of "read-

only" operations can should be developed to create a sort of pre-flight validation. These could include prerequisite checks, property populations, or system specific actions.

At this point the Publisher can query the systems in much the same way by executing a "GIVE ME STATUS" message to the intended systems. This message may look like the message below:



Retrieving the status should be as near to instantaneous as possible. This is a process that can be called either "interrogating the execution" or "fingerprinting the status". This can be done in a variety of ways, and will most likely be very specific to the process that was executed in the previous command. The status returned will likely be of a few common conditions; RUNNING, EXITED SUCCESSFUL, EXITED FAILED. The operator or "orchestrator code" communicating through the Publisher will investigate the statuses from all 10,000 systems. One would imagine that once all 10,000 systems return EXITED SUCCESSFUL the operation can be considered complete, however given the number of systems it will also give the operator or orchestrator the knowledge of the exact number (and identification) of systems that FAILED and allow corrective actions to commence, either orchestrated through the Publisher or otherwise. For example the INTENDED AUDIENCE of the messages could be modified to be inclusive of only FAILED systems.

***Operational Case 2 – Datacenter Power Failure**: The fleet has 50,000 systems of varying types; 5,000 database servers, 25,000 application servers, 10,000 webservers, and 5,000 batch processing systems. The datacenter has lost power and is running on diesel. You have 30 minutes to bring all systems and services down in their proper order in order to avoid data loss and corruption. Specifically, webservers must come down first, followed by the application servers and batch processing servers, and then all of the database systems.*

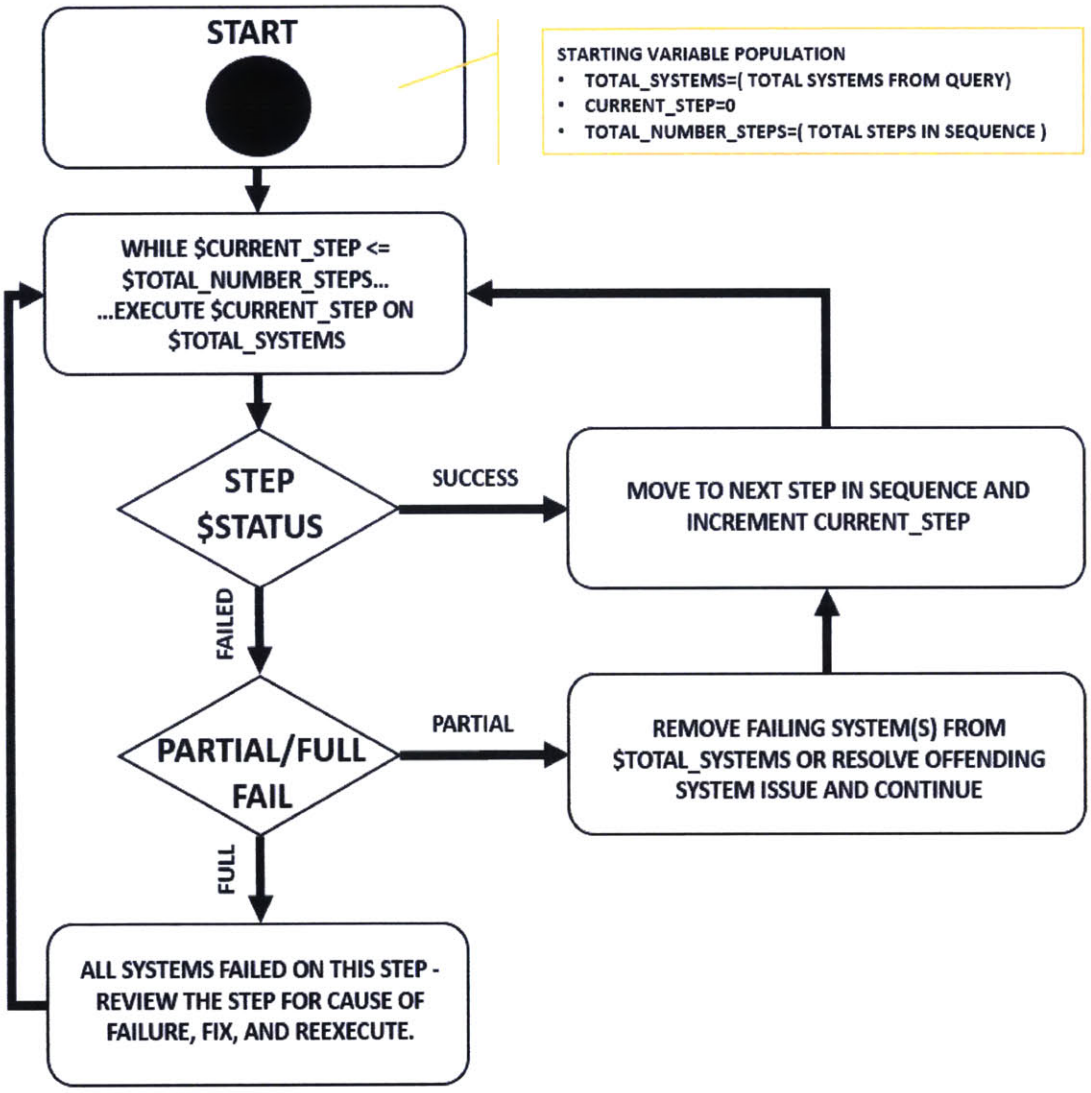
Without shortcutting too much, the discussion around these cases will be limited to the actual messages placed out on the "execution topics." This case 2 will be performed by publishing the following messages with the following contents:

Table 8 - Case 2 Operational Message Sequence

MESSAGE NUMBER	INTENDED AUDIENCE	CONTENTS
1	ALL SYSTEMS WHERE ATTRIBUTE sysFunction=webserver	RUN "Shutdown webserver instance" and send to background & PRINT "SHUTTING DOWN WEBSERVER"
2	ALL SYSTEMS WHERE ATTRIBUTE sysFunction=webserver	QUERY $STATUS OF webserver AND PRINT "WEBSERVER $STATUS"
3	ALL SYSTEMS WEHRE ATTRIBUTE sysFunction=application OR batch*	RUN "Shutdown database instance" and send to background & PRINT "SHUTTING DOWN APPLICATION AND BATCH SERVER"
4	ALL SYSTEMS WEHRE ATTRIBUTE sysFunction=application OR batch*	QUERY $STATUS of application or batch server AND PRINT "APP or BATCH $STATUS"
5	ALL SYSTEMS WHERE ATTRIBUTE sysFunction=database	RUN "Shutdown database instance" and send to background & PRINT "SHUTTING DOWN DATABASE SERVER"
6	ALL SYSTEMS WHERE ATTRIBUTE sysFunction=database	QUERY $STATUS of database server AND PRINT "DATABASE $STATUS"
7	ALL SYSTEMS WHERE ATTRIBUTE sysFunction=webserver OR application OR batch OR database	RUN "Halt System" PRINT "HALTING"

NOTE: The queried $STATUS indicates that that the status of the system (or applications being queried) is variable.

The Actions 1-6 described above are a series of execution/status couplets that are done against a variable set of systems, specifically the set of Subscribers that respond to the publishers query. This sequence of events is shown below:



The intention of the diagram above is to simply give the flavor of what an operator would consider when executing arbitrary commands to any variety of systems in the fleet. For example Step 1 and 2 above should expect 10,000 systems to respond to each step. Step 2 would can be executed repeatedly until such time that the fully response indicates 10,000 SUCCESSFUL or FAILED responses. Anything less will be cause for decisions to be made by the operator or the orchestrator. For example, If all 10,000 webserver systems responded back with 9,950 SUCCESS and 50 FAILED then the operator or orchestrator will need to decide how to handle the remaining 50 FAILED systems. At this point the systems can either be fixed or corrected, or removed from the $TOTAL_SYSTEMS to move on with steps 3 and 4. This approach could be described as a "Lockstep" ITPA and orchestration approach, and though there are a great many benefits

to using this approach there are also some pitfalls that will be addressed in the "Pitfalls of the Lockstep Approach" section below.

Operational Case 3 – Change Stage Only: *The same fleet of 50,000 systems comprises a set of 10,000 customer environments. Maintenance is approved only for stage systems and services (half the fleet) and will be inclusive of the following tasks:*

1.) *Shutdown the webserver processes on all "Stage" webservers*
2.) *Run a prescribed set of update steps on all "Stage" application servers. EX deploy new code to enterprise applications running within the application server containers.*
3.) *Restart all running "Stage" application server processes.*
4.) *Start the webservers processes on all "Stage" webservers restoring services.*

In this case to avoid unnecessary redundancy we will simply mention only what is the intent of this case; to illustrate the process of combining attributes in the messages in order to further refine which system will respond to the messages. In particular, steps 1-4, each require only the "Stage" systems to respond to actions called for in a message. This is particularly important to understand in this framework. Adding additional attributes to the identity section of the message gives the ability to create the logical AND necessary to define groups that are highly specific. In this case the sysFunction will continue to be defined as "webserver", "application", etc but a new attribute will be added to further refine the grouping. In this case we will have an attribute called sysPhase, which can be Development, Stage, or Production. Our query in Case 3 will be to only systems with the attribute sysPhase="Stage". Keep in mind that an attribute is really just a specific system level variable defined by the fleet or datacenter system's architect. In most situations the attributes are modifiable to a given state of the systems or applications therein. One can employ different strategies to deal with how modifications are made but it is best to have a known set of attributes at the time the systems are brought online. This is particularly true if the publish subscribe messaging pattern is used as part of a provisioning cycle.

Pitfalls of the Lockstep Approach

Despite the numerous advantages of using a lockstep approach for orchestrating any variety of system or application level tasks across a given Service Providers datacenter it is not a silver bullet. For the lockstep approach to work perfectly, it would require that every execution against every system acted near identically for homogenous system types if timing is truly of the essence. This is rarely the case. As an example and going back to the Operational Case 2 above, what would happen if the timing around a given

execution fell along a standard bell distribution curve for steps 1, 3, and 5? The completion times of the command execution would show that 95% of the systems will complete their given execution within 2 standard deviations of the mean. This is likely fine if the standard deviation is small. But what will happen if the mean execution time is 10 minutes with a standard deviation of 3 minutes. In short 50% of the systems will need to wait around idle and in a completed state for at least 9-10 additional minutes before being able to move onto the next step. (And that is only if the remaining systems come in successfully) It is highly possible that the time constraints to complete the given set of steps will not accommodate this type of lockstep approach. But is there a way to develop this publish-subscribe messaging pattern in order to orchestrate multiple steps within a single "system or application groups" independently from any other "system or application group" and still have them operate in parallel? Particularly when the goal is to allow each group to complete as fast as it can without regard to any other group. To accomplish this through a single publisher/broker is highly difficult particularly when the expectation is to scale to thousands or hundreds of thousands of "system or application groups" perhaps involving millions of individual systems.

Without even considering the incredibly complex messaging and execution logic that will ultimately be associated with this kind of solution from a single publisher/broker to multi-simultaneous workflow solution, the architect really must consider the messaging load that will undoubtedly be necessary to facilitate all of the necessary status checks. For example; if only 10,000 systems were involved in the multi-step ITPA workflow, the orchestrator would need to publish highly targeted messages to each of those 10,000 systems almost defeating the purpose of a broadcast solution. To address this core problem the ITPA automation solution we must begin to develop the concept of a "Micro-Orchestrator" described in the following section.

Automation via a Micro-Orchestrator

The previous section discussed some of the architectures, strategies, benefits, and pitfalls of using the Publish-Subscribe messaging schemes to horizontally orchestrate and automate ITPA based tasks in very large fleets. The size of and members of these self-forming groups are generally limited to the capability and performance limitations of the network and/or the system housing the message broker. But a system fleet can be managed well into the hundreds of thousands using this scheme and makes it an ideal solution for fleet wide automation and orchestration. However, when one tries to apply conditional logic to "daisy-chain" IT processes between *and* across multiple "groups of groups" is where this single publisher/broker

messaging scheme begins to show some weaknesses. This is particularly true when "first to finish" and timing are key. To overcome these weaknesses let's introduce the concept of a "Micro-Orchestrator."

The Micro-Orchestrator is the same thing as the full-fleet Publish-Subscribe paradigm but flipped 90° and localized to the group. The purpose of the Micro-Orchestrator is not to replace the fleet-wide orchestration scheme, but rather to handle the flow-control of ITPA tasks within a given application group or "stack". Now instead of each system being only a subscriber to the full fleet's "execution" topic there will now be an additional "execution" topic specific to the group. Each system in the fleet will now listen to two "masters"; the fleet wide publisher, and the group wide publisher (i.e. the Micro-Orchestrator).

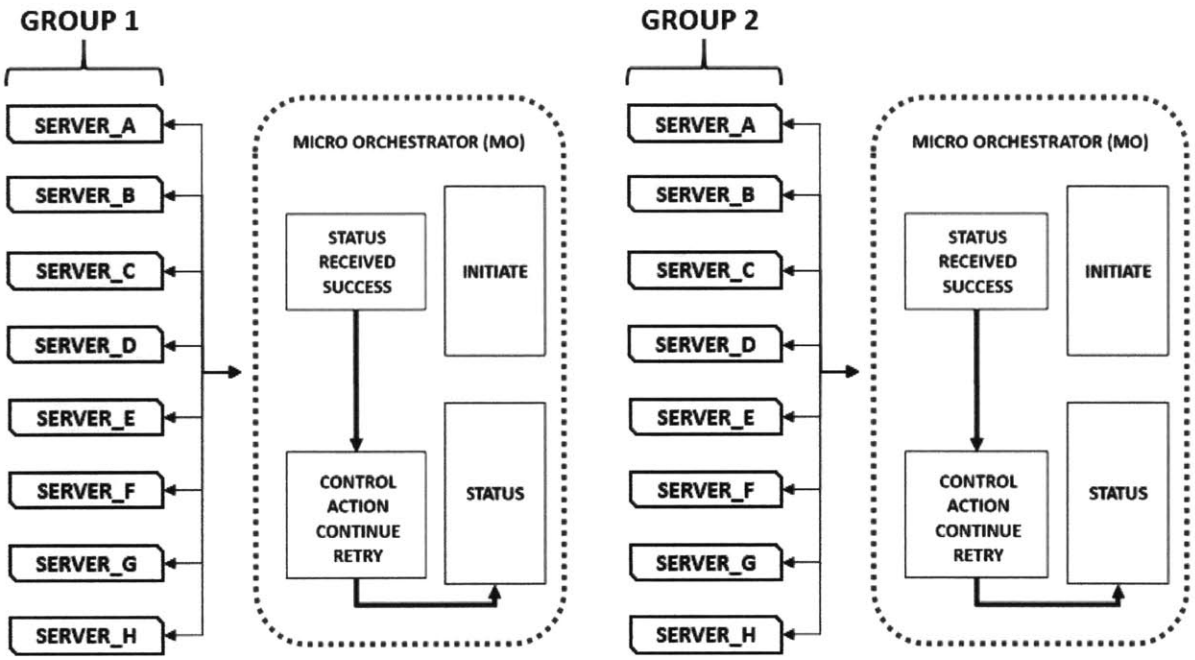


Figure 20 - Group Based Micro-Orchestrator

The Micro-Orchestrator will have no visibility or knowledge of any other subscribers, publishers, or brokers in the fleet other than the ones within its group. In other words it is exclusive to the group. The Micro-Orchestrator will at a minimum be responsible for the following higher order ITPA tasks.

1.) Initiation of group based process workflows.
2.) Receipt of the Status of a given process (real time or queried)
3.) Flow control between ITPA tasks between the systems in the group.

4.) System Addressing or Querying scheme.

To illustrate the micro-orchestrator let's revive the 7 process workflow involving tasks A-G so that we can follow the control loops around to see how micro-orchestrator must handle the execution and control actions in response to statuses it receives from the micro orchestrator Subscriber agents. The following is a more generic control diagram of the "External Orchestrator" shown in the previous section.

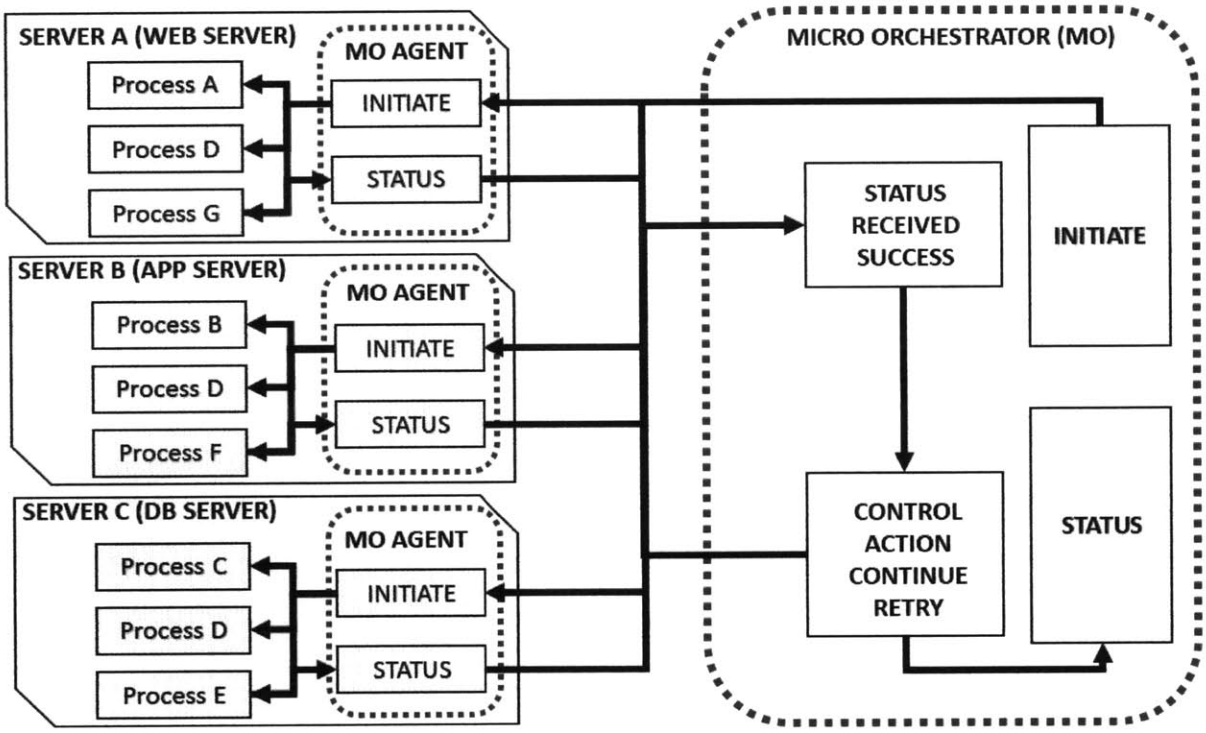


Figure 21 - The "Micro Orchestrator"

Using our Publish-Subscribe pattern, the Micro-Orchestrator is the "Publisher/Broker" and the MO-Agent is the Subscriber. Below is a description of some of the interactions between the Micro-Orchestrator and the Orchestrator Agent shown above in the in 7 Process workflow. Although the actual logic and methods used between the Micro-Orchestrator and the MO Agents will vary considerably depending on the workflow being defined, below is a possible approach:

1.) The overall 7 Process workflow is initiated by the Micro-Orchestrator.
2.) The Micro-Orchestrator passes a message to the Micro-Orchestrator Agent on Server A informing Server A to initiate Process A.
3.) The Micro-orchestrator must then receive the initiation status of Process A. (Did Process A start?)

a. Because it is unknown in this workflow how long Process A will take to complete the micro orchestrator need only know that the agent actually initiated Process A and nothing more.
b. The micro orchestrator agent will immediately pass back a message to the micro orchestrator indicating the status of the initiation only, for example: initiated (SUCCESS) or not initiated (FAIL).
c. The way a particular success or failure of a process initiation looks is variable depending on the operating system and many other factors. However, the logic to derive the status of an initiation is at the Server level and then translated within the "message" passed back to the micro orchestrator and into to something understandable by the workflow logic. The contents and type of messaging will be determined by the orchestrator, agent, OS, and workflow logic.
d. For the purposes of this discussion we will assume that Process A is initiated successfully and that the "success" message is sent back to the orchestrator.

4.) Following the successful initiation of Process A the micro orchestrator will poll the micro orchestrator agent for the status of Process A.
   a. This polling can take many forms but in general it is meant to find out what the state of Process A is at the time of the poll. Some simple examples of Process A status could be
      i. RUNNING
      ii. SUCCESSFUL
      iii. FAILURE (PARTIAL OR OTHERWISE)
   b. The workflow logic will be responsible for defining the frequency, number of iterations, and the timeouts associated with the status polling of any given process.
   c. The workflow logic will also determine what happens "next" when a status is received. For our purposes we will assume that it will exit on FAILURE, iterate on RUNNING, and continue the workflow on SUCCESS.
   d. For all intents and purposes the method to "poll" and retrieve a status will look identical to the method used to "initiate" a process from a functional perspective. For example; to get a status is pretty much the same as the initiation of a process, with the exception that process being initiated simply gets the status of the previous process instead of initializing something new. Again this can have many forms.
   e. In this example we will assume that Process A completes SUCCESSFUL after 5 minutes and the micro orchestrator retrieves this status from the micro orchestrator agent.

5.) Once the status of Process A is received by the micro orchestrator the micro-orchestrator will send a message to Server B to initiate Process B.

From this point forward the 7 process workflow will loop around initiating processes and receiving statuses of those processes until the workflow either completes or fails.

The previous illustration shows how workflows can be orchestrated in and intra-group way. But clearly this is limited in application. The key to a robust ITPA solution for very large IT Service providers is to scale. Now, at this point in the discussion, we have the raw materials to create a framework that can approach the ITPA and orchestration scheme from any angle:

- Very large groups of varying vertical complexity
- Very large groups of varying horizontal complexity
- Very large groups with varying levels of both horizontal and vertical complexity.

This can be done by combining the horizontal fleet wide Publish-Subscribe messaging scheme to handle horizontal complexity and scale, with the Micro-Orchestrator's Publish-Subscribe messaging scheme to handle vertical complexity and scale. To illustrate this the concept of the "Macro-Orchestrator" is now introduced.

Automation via a Macro-Orchestrator

This is an important point for the ITPA solution being developed in this paper. With the use of the Micro-Orchestrator we are able develop complex workflow automations that will control the conditional movement of processes within a group of logically linked systems at any level of vertical complexity. With the single Publish-Subscribe "Horizontal" automation and orchestration scheme we are given the breath and scale to target execution to any given system or group of systems in the fleet.

From the perspective of the central Publish-Subscribe "Horizontal" automation and orchestration scheme the system that runs Micro-Orchestrator is no different than any other individual "Subscriber" system in the fleet and therefore can be queried for execution in the same manner as any other system. The central Publish-Subscribe Publisher/Broker will now become the "Macro-Orchestrator"

The diagram below illustrates the role of the Macro-Orchestrator, which now serves two roles.

1.) Horizontal "Group-Based" automation and orchestration, and
2.) Macro "controller" of multiple Micro-Orchestrator Controllers (i.e. the Macro-Orchestrator)

With this new connection between the two, the Macro-Orchestrator can now initiate any given Micro-Orchestrator ITPA workflow by making a single call to a group's Micro-Orchestrator and querying it for status in the same way as any other Process Execution-Status copulate in the fleet. This moves the conditional logic and overwhelming message load of complex workflows out of the domain of the Macro (central) – Orchestrator and leaves it squarely within the control of the Micro-Orchestrator. See below:

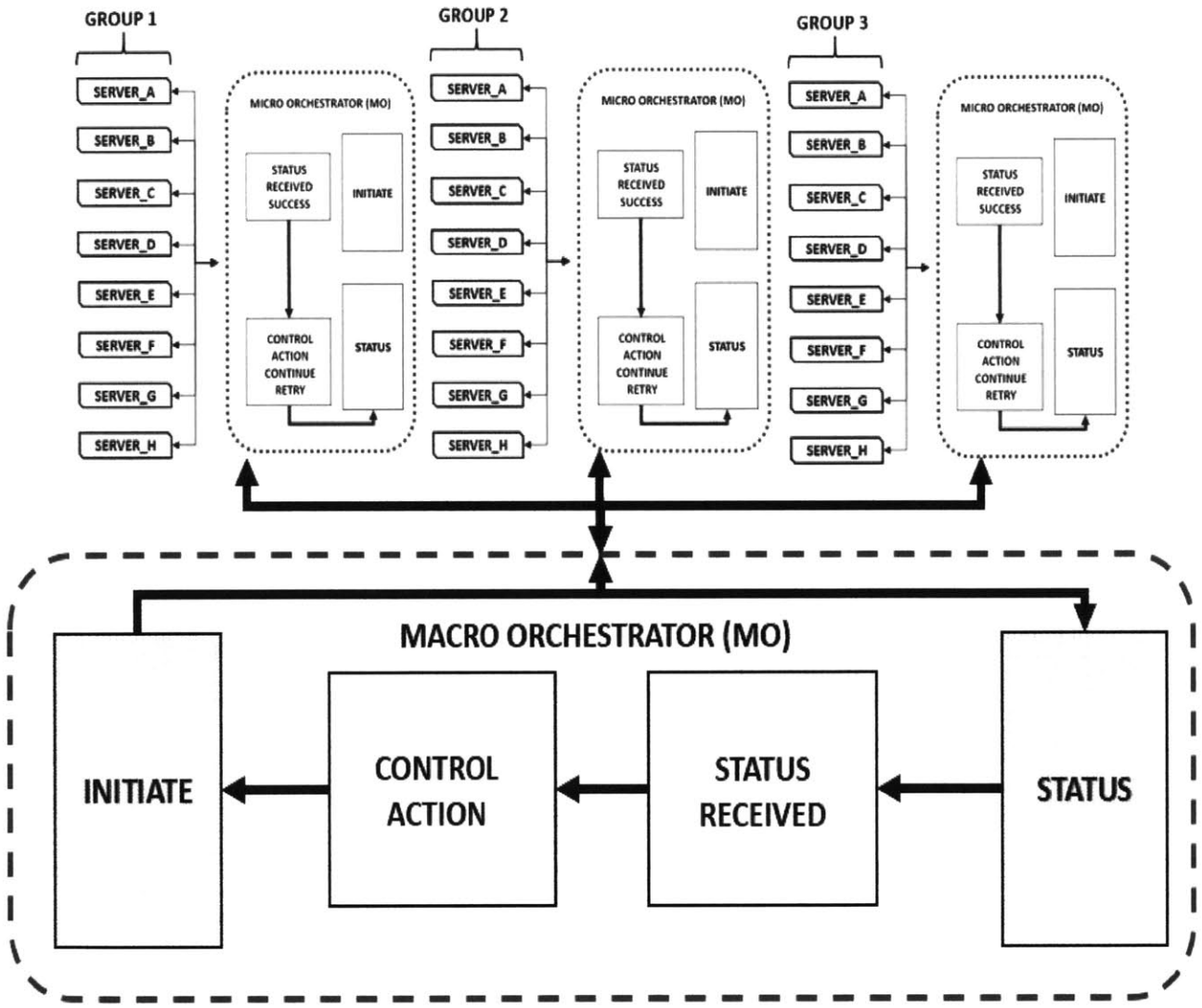


Figure 22 - The Macro Orchestrator - High Level Design

This new method allows complex intra-group workflows to move completely away from a lockstep orchestration scheme and solves both of the Horizontal "group based" orchestration pitfalls. Lockstep timing and conditional messaging load. Now with the use of the Macro/Micro-Orchestrator combination

the first and fastest system to complete its individual ITPA task in a given group based workflow will no longer be tied to the slowest system of the slowest group to finish its operations before moving to the next task in the workflow.

Revisiting our 7 Process workflow, now a single group's full workflow can be viewed as a "single" task from the Macro-Orchestrators perspective. That single task can have the same status types as a single systems task, SUCCESS, RUNNING, or FAILURE. (Amongst many others)

Though the illustration below only shows 2 "groups" of systems being "orchestrated" by the Macro-Micro Orchestrator combination, this same framework can be used to orchestrate any number of groups. It is certainly possible to have 10's of thousands of "groups" controlled by the Macro-Orchestrator and by extension many hundreds of thousands (if not millions) of individual systems controlled by their individual group's Micro-Orchestrators.

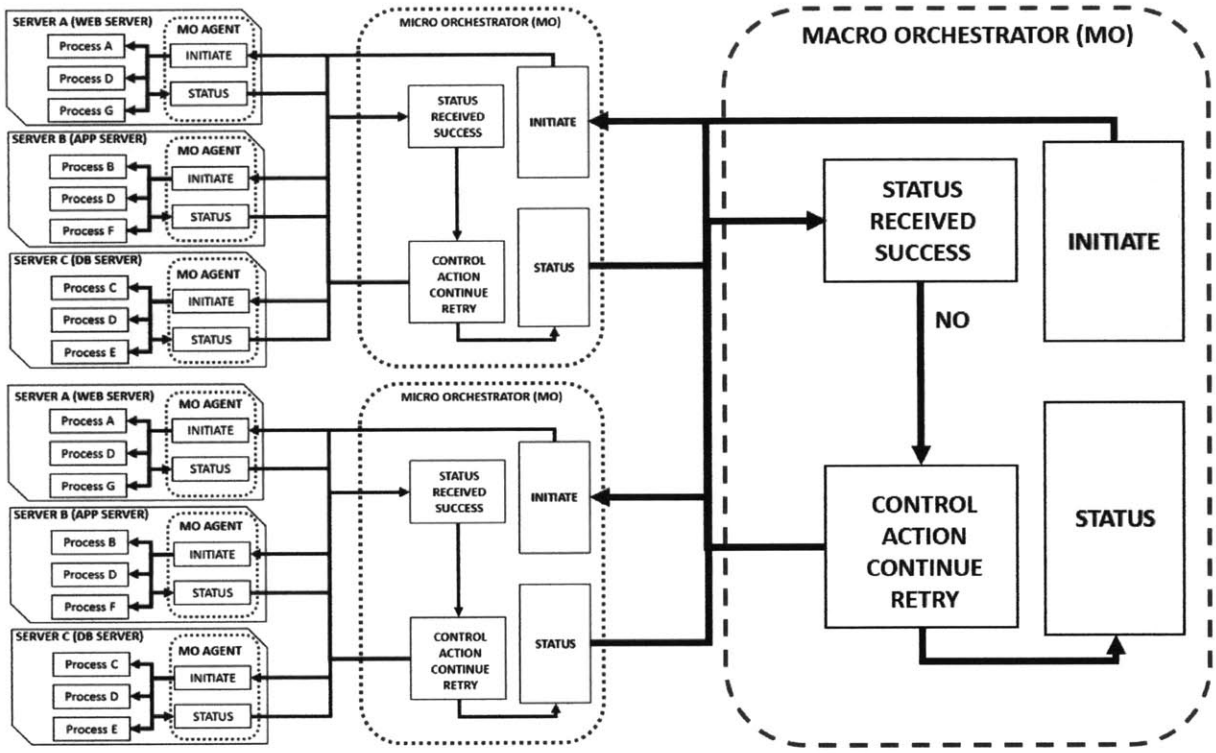


Figure 23 - The "Macro Orchestrator"

This approach does not need to be limited to 2 tiers of orchestrators either. In fact, the lightweight nature of a subscriber makes it possible to create any number of "zones" of orchestration within a very large fleet of systems. Perhaps the most interesting fact of the Micro-Macro Orchestration approach as that it can tackle ITPA automation, orchestration, and workflow schemes across system fleets with almost any level

of vertical or horizontal complexity. It can also fit into almost any system or datacenter architecture, with any type of tenancy model, and in organizations with almost any level of process maturity. It is also abstract enough that there are a great variety of open source software products which can be used to create this type of ITPA solution without being unnecessarily bound by a particular software vendor, or product suite.

CHAPTER THREE – DESIGN CONSIDERATIONS AND BEST PRACTICES

Understanding Safety and Managing Change through ITPA

If there are two laws that demand the respect of any large scale IT Service Provider they are undoubtedly Moore's Law and Murphy's Law:

Moore's Law states that the number of transistors on an integrated circuit will double every two years until the limits of physics are hit. This has been abstracted to a more general law of "exponentials" which seems to indicate a doubling in the processing and capacity of computer hardware during this same period. (Moore, 1965) This observation has accurately predicted the exponential growth in compute capacity over the past several decades and is expected to continue on this same trajectory into the near future. Perhaps as a bit of hyperbole it could be stated that if this trend were to continue on its current path, the most powerful computer today will be approximately 33.5 Million times slower than the most powerful computer available 50 years from now and 1 Billion times slower in 60 years. There are reasonable arguments to be made about the future plausibility of this growth continuing or plateauing, which is not the intent of this paper to describe, but rather to state; that at present, the exponential growth continues in this regard. Perhaps as the possibility of "quantum computing" matures the continued acceleration in compute power will continue to be seen for many generations to come.

Murphy's Law states that anything that can go wrong, will go wrong.

So what do these two "laws" have to do with how Service Providers provide services? For starters, the doubling of compute capacity every couple of years has the effect of creating a strong negative correlation between the Service Provider's ability to grow and its probability of obsolescence. Meaning, if there is little to no growth of the Service Provider or its services, the probability of its obsolescence is pretty high. This implies change, a lot of it, and fast. Therefore, Murphy's Law in this context, is an invitation for calamity and failures as changes are introduced.

The illustration below is a simplified dynamics model displaying the relationship between growth, obsolescence, and change.

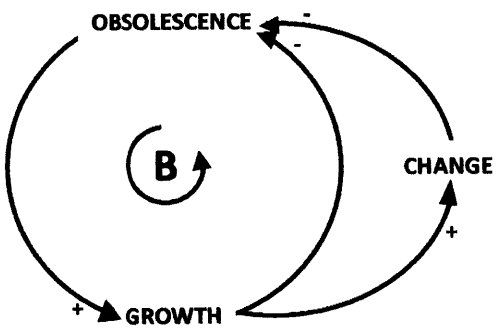


Figure 24 - Relationship between Change, Growth, and Obsolescence

If everything remained standardized and nothing changed from the time systems and applications were provisioned, life for the IT Service Provider would be easy. This is simply not reality though, and change happens very frequently. Even with the deployment of a standardized service environment there is always the inevitable drift that will shift any given environment from highly-standard to highly-modified. This is particularly true for emerging technologies and for services that require a high level of customizations to meet customer requirements. Change and growth, however, will always be necessary for any service provider to avoid service obsolesce. For example if an organization's services are becoming obsolete in the market place, they must grow or change along one of many service dimensions which include: performance, functionality, cost, availability, or resiliency.

"Safety"- an emergent property of complex systems, is particularly important to consider when change is unavoidable and frequent. Inevitably an ITPA solution will be used to introduce changes to the service infrastructure. Therefore, the safety of the ITPA control structures should not just be evaluated by looking only at the "last event in the chain" that might cause a failure. One needs to take a holistic view of the entire "system" being orchestrated and automated.

Failures can come from any number of intrinsic and extrinsic factors to the systems being automated. In an influential paper by Nancy Leveson, titled "A New Accident Model for Engineering Safer Systems" (Leveson, 2011) she states that there are several conditions which lead to an unsafe system and that it is those conditions without proper controls that ultimately leads to system failures. The operating conditions of these failures generally go well beyond sequential events proximal to the failure. They include:

- Social and Organizational Factors

- System Accidents and Software Errors
- Human Error
- Adaptation
- Emergence and Constraints.

Leveson states that "accidents occur when external disturbances, component failures, or dysfunctional interactions among system components are not adequately handled by the control system, that is, they result from inadequate control or enforcement of safety-related constraints on the development, design, and operation of the system. Safety then can be viewed as a control problem, and safety is managed by a control structure embedded in an adaptive socio-technical system. The goal of the control structure is to enforce constraints on system development (including both the development process itself and the resulting system design) and on system operation that result in safe behavior." (Leveson, 2011) Development of an ITPA control structure and automation scheme, particularly one with the breath and reach of the type mentioned in this paper, demand very careful design to avoid unforeseen and unnecessary system failures.

Imagine a circumstance where a message (as described in Chapter 2) is accidently addressed to ALL systems listening on a particular topic. However, the message was actually intended for a very small group of systems but was misdirected due to the message author's confusion about how the "regular expression" would expand. This is human error. Now imagine that the "action" taken was a one-off update to modify a critical and untested system function. The plan was to thoroughly regression test this modification against the very small group of systems, garner management approval for the modification, and ultimately push this modification to the remaining systems in the fleet. Now, however, you have just accidently released this change, untested, to the entire fleet of systems. There is a possibility that the modification will be innocuous and all will be good. There is also a possibility that the modification will cause a full failure of every service in the fleet. Catastrophe, particularly if there are strong SLA's in place, or the systems are uptime critical. Clearly no one would not be so haphazard about the introduction of new code to the entire systems fleet, yet somehow this type of event seems to happen regularly regardless of the process used to implement the change. The regularity of this type of failure event is why it is vital that "controls" be implemented to prevent hazards like this from occurring.

A highly simple example of a control that can be put in place for a given operation is to use the concept of a rifle's "safety-catch." In most forms the safety-catch will only allow the trigger of the rifle to be pulled if the safety-catch is first disengaged or turned off so it is no longer blocking the trigger from being pulled.

This mitigates the possibility that the trigger will be accidently pulled and the weapon discharged. If the logic of the action being executed through the ITPA framework first checks for the engagement of a "virtual" safety-catch and the default position is "safety-catch=on" then the potentially catastrophe event can be averted. In order to enable a system for execution through the ITPA framework, there should be a decoupled execution of a toggle that will switch the safety-catch from "on" to "off" before any action can be executed.

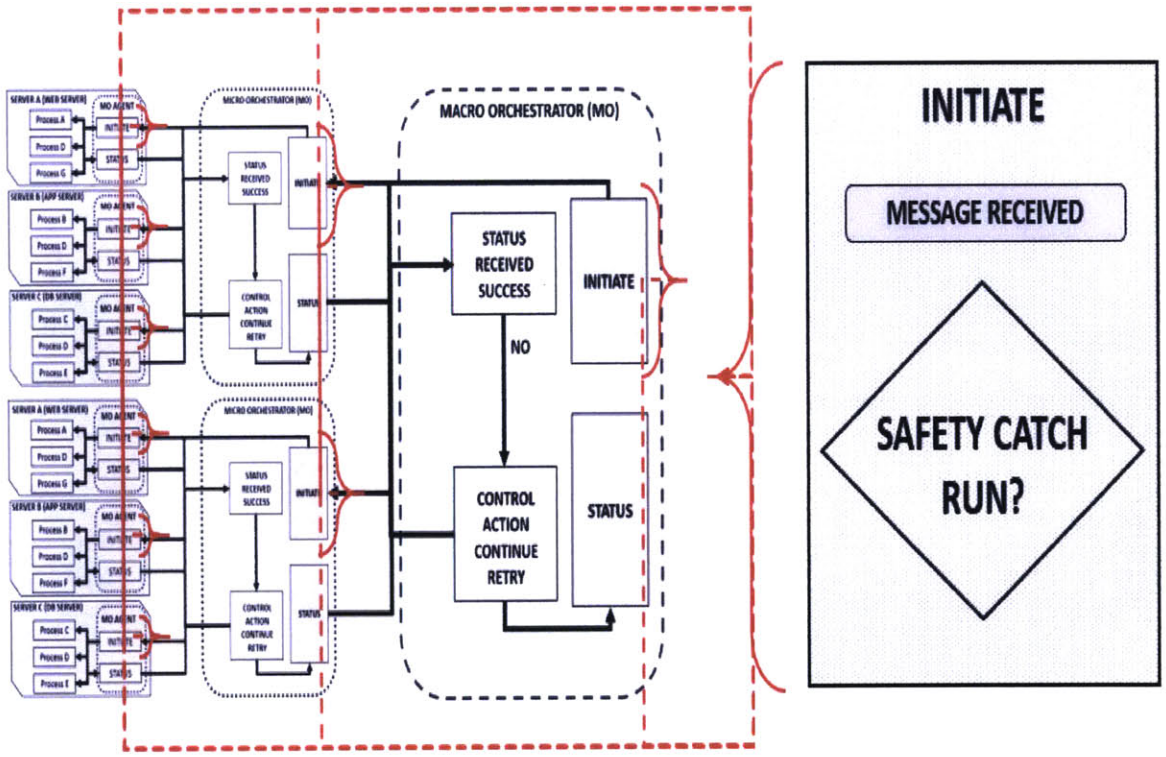


Figure 25 - Possible "Safety Catch" implementation

A virtual safety-catch can take many forms and can be implemented in any variety of ways, but it is recommended that these types of default safety controls be implemented at the onset of a given ITPA solution. This safety catch should be considered as a "security" catch point, including the ability to pass and interrogate public private key information, passwords, or any variety of other security and sanity checks. They should also be layered throughout the ITPA workflow and process execution control structures to prevent unintended execution at several strategic points. Designing these types of controls adequately will ultimately lead to a "safer system." Each workflow or process automation developed will

ultimately be unique, so the implementation of the safety controls will also be unique. The ITPA solution in this paper gives the necessary flexibility to the process engineer or workflow developer to design safety controls, which match the particular needs of the IT Service provider and service being offered. The controls and virtual safety-catch mechanisms should also not be limited to just the code. These types of controls should be thought through and implemented at the organizational level as well.

Clearly a safe ITPA solution is of very little value if it is not useful. The usefulness of it will be determined by how well understood it is by the implementing IT organization. For example, granting a large organization the ability to create workflows and automation solutions has a great many obvious benefits. The drawbacks are also obvious, particularly where the workflows and automation solutions involve change and configuration management. (Which they undoubtedly will, unless the Micro-Marco Orchestrator scheme is only used for reporting purposes.)

Another design consideration related to safety is the integrity and reliability of what is being executed. For example, suppose the message above *was* truly intended to the entire fleet of systems and the action to be executed was "**runUpdate**." What would happen if the runUpdate action was inconsistent from one system to another, but it was supposed to be identical? What if System A's runUpdate would perform tasks 1,2 and 3, while System B's runUpdate still only ran task 1 and 2? Even if runUpdate completed "SUCCESSFUL" on both systems, it would create an inconsistent state between System A and B because A will have tasks 1, 2, and 3 completed, and system B only one and 2. This condition would not be easily caught particularly if the orchestrator was only checking for the overall status of the full execution of runUpdate and not the subtasks contained within. Perhaps the easiest way to deal with this is to keep a single shared action repository for the fleet. It is also possible to derive some type of reconciliation scheme that will keep the repository equivalent between system A and B despite the fact that they are different copies of the same repository. The latter is a must less desirable approach due to the complexities of synchronization across very large numbers of systems. None the less, the same ITPA solution described in this paper could also (and perhaps recursively) be used to maintain a synchronized execution repository for the fleet. The distinction between a single command repository and a synchronized command repository is shown below.

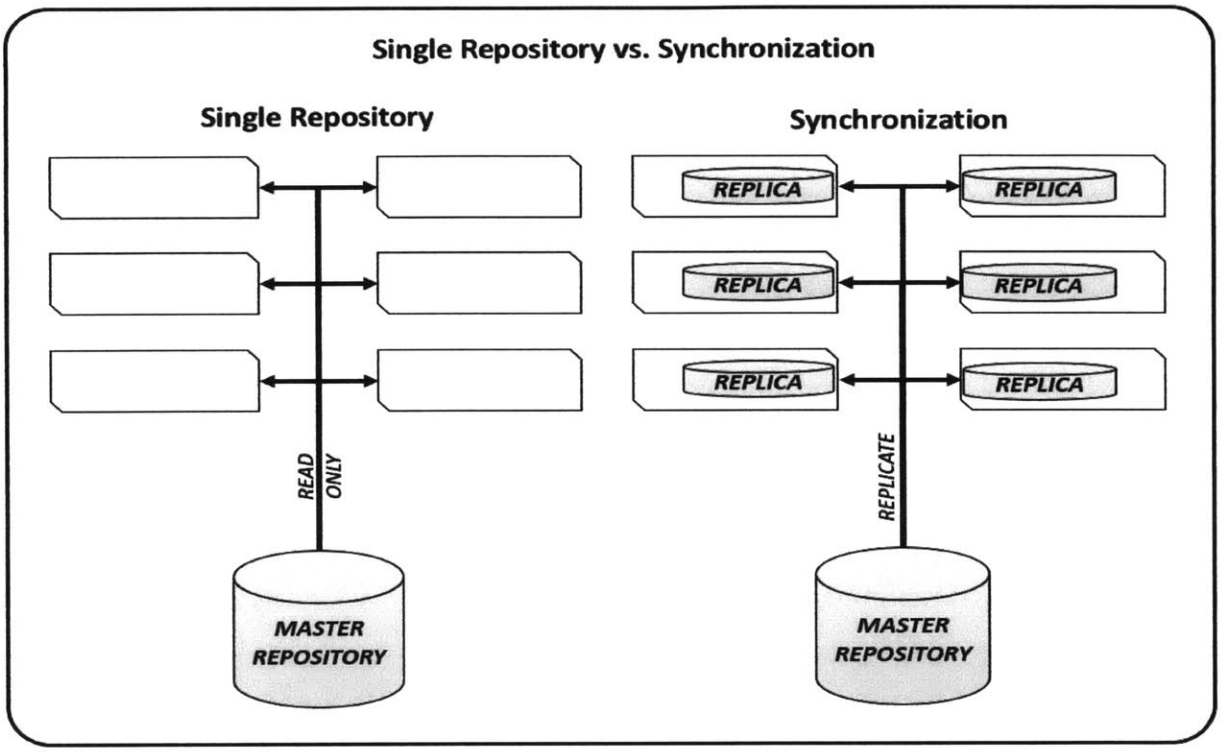


Figure 26 - Single Command Repository vs Synchronized Command Repository

Two other considerations when creating workflows or process executions within this paper's ITPA framework are described below:

1.) Create a method that will guarantee that the invocation of a specific process (command execution) has occurred or not occurred. So couple the status of the execution (did it start) and the actual execution into a single SEND-ACKNOWLEDGE coupling (see STOMP and our implied send-reply communication between the publisher and subscriber in Chapter 2)
2.) Decouple the status of *what* was executed from its actual execution. Particularly for long running processes.

There's an important distinction between the statuses above. One answers the question, "Did it start?" the second answers the question "How did it finish?"

This somewhat minimum criteria should be considered when creating the statuses and messaging between the orchestrators and the subscribers that are part of the same "topic" domain (both macro and micro). In particular, if the subscriber replies back that it executed, it is vitally important that it did indeed execute. This will give confidence to the operator or the orchestrator code author that what was reported to have happened is actually what happened. An example could be the following:

A message is directed to the systems containing the desired attributes and those systems call a command wrapper that executes in the following in order.

a.) Execute the intended process, detach from it, and background it.

b.) Print a message that the previous line executed and encapsulate that message in the reply message to the publisher.

Both (a) and (b) are executions and they must occur in sequence, (a) then (b). Therefore the only way to get to (b) is to have completed (a). If (a) is sent to the background then (b) will simply report back via a reply message that (a) has run. This assures the receiver of the reply message that (a) actually occurred else they would never have received the message printed by (b). This coupling has an added benefit as well; the operator or the workflow orchestration code should not need to wait until the command is fully executed in order to know it was actually executed or wait to receive the exit status of the executing process in order to know it started. This is particularly important when processes that are invokes take several minutes to several hours to complete.

The decoupling of the status in number 2 above is also an important feature of any messaging flow. If the execution is decoupled from the status, then the publisher or orchestrator can always expect an instantaneous (or near instantaneous) reply message from the executing subscriber system. In Macro-orchestrator based execution this will allow the operator to quickly view the status in a real time way to a very large set of systems. If the status execution is from the command line interface (say on a Linux system), or collected in an array through an orchestration mechanism, the results can easily be parsed through using standard UNIX tools or programming languages, or visually inspected for patterns indicating success, failure, or processing. This also allows the micro-orchestrator to continually poll systems for status and allow for multi-threading techniques without being held up on long running process executions.

Of course there are truly an unlimited variety of ways that an IT Service Provider could use this publish subscribe based ITPA framework to manage automation and orchestration requirements. The flexibility of this ITPA framework is very high and somewhat limited by the maturity of the IT organization implementing automation and orchestration through it. There are times, however, where IT process automation is not the best course of action and can itself lead to an unsafe system state. Some of these pitfalls are described below.

Pitfalls and Failures of Automation

Automation is essential to any modern IT organization, however it is not always appropriate or desirable for every circumstance. There are several instances where automating IT processes will only achieve minimal or even negative net results. This, even if the automation is exceptionally well done. Automating without evaluating the benefits and risks of the automation can lead to a high degree of organizational "wheel spinning." In other words, teams may spend a significant amount of time and effort automating processes that were better left manual. It's also possible that the actual automation will lead to a high degree of downstream (or upstream) dysfunction. For example, automating a provisioning process in such a way that makes operations, maintenance, and support of the service more complicated will have a net negative impact on the entire service organization. What is an efficiency win for provisioning is a loss for others. Therefore, when evaluating a particular process automation, one must look not just at the immediate impact of automation but also on the upstream and downstream impacts of the automation.

In a book titled *Aviation Automation: The search for A Human-centered Approach* (Billings, 1997), Billings explores some of the costs and benefits of aviation automation. Though Aviation and IT are not identical topics, from the perspective of automation, they share a remarkable number of commonalities. This is particularly true in the way that that IT Operators staff regularly interact with complex IT systems in a manner quite similar to a Pilot's interactions with their aircraft and flight control systems. Billings sites the following from the 1989 ATA Human Factors Task Force:

> *"During the 1970s and early 1980s, the concept of automating as much as possible was considered appropriate. The expected benefits were a reduction in pilot workload and increased safety...Although many of these benefits have been realized, serious questions have arisen and incidents/accidents have occurred which question the underlying assumptions that the maximum available automation is always appropriate, or that we understand how to design automated systems so that they are fully compatible with the capabilities and limitations of the humans in the system"* (Billings, 1997)

The author goes on to state design factors that can lead to a high degree of confusion for operators of "the system." This confusion can ultimately lead to multiple failure scenarios. In the case of Billing's book "the system" is the aircraft, the flight control mechanisms, and human to machine interactions therein. Below there are several of the "design flaws" illustrated by Billings which have been abstracted and given in the context of IT systems. The general spirit of Billing's cautions for each of these categories is still intact. These can serve as general cautions and considerations for any IT automation solution.

Failures of automation systems often times will result at the design phase. When accidents occur however they are typically investigated from a "chain-link" vantage point. For example, instead of looking at the design flaws of the system the investigation for root cause will typically focus on how this event lead to that event, which ultimately caused the accident. A more holistic view must be taken when creating the automations and automation systems in order to avoid conditions which would not otherwise exist if careful consideration was taken with regards to the below design factors that can lead to difficulty.

Automation Design Factors that Lead to Difficulty Modified from Billings for IT examples (Billings, 1997)

Design Factor	***Difficulty***
Complexity	Complexity (particularly complexity hiding) is a consideration that needs to be understood. Particularly where automation components interact with other systems automated components. Without understanding the different interactions it's possible to create a considerable amount of confusion when certain interactions default to unknown modes and create undesirable circumstances. Controlling interfaces and interactions between the system being automated and external systems is key to good design.
Brittleness	As Billings describes it "Brittleness is an attribute of a system that works well under normal or usual conditions but does not have desired behavior at or close to some margin of its operating envelope." These types of conditions exist in many forms and are difficult to test for, even under extreme testing circumstances. The operating "margins" should be reviewed when constructing any ITPA automation system or framework.
Reliance on Automation	As automation is used and implemented there will ultimately be some level reliance attributed to it. This lever didn't cause the light to go off but was assumed to work properly, and continuation to the next step continues. This reliance or faith in the automation system can lead to numerous failure conditions. This is not necessarily a design consideration per se but should be considered as something to be addressed in the ITPA solution.
Clumsy Automation	This one seems somewhat self-evident, but Billings describes as a "descriptor to denote automation that lightens crew workload when it is

	already low, but requires more attention and interaction at times when workload is already high"
Data Overload	The data that can emerge through automation, though extremely valuable, can ultimately lead to distraction from other areas where attention should be given, or rather consume so much time interpreting the inflow of data that the other areas that require attention may be missed.
Skill Degradation	This again is not entirely design flaw, but certainly a consideration to be taken into account during the design phase. Billings specifically states that "one potentially serious problem in human-machine systems with highly capable automation is a loss of certain skills by the human when the automation routinely performs tasks that require such skills" Automation is not a replacement for necessary skills and it should not be assumed that an operator with the capability to run a particular automation has the necessary skills to perform the underlying automation.

Table 9 - Automation Design Factors that Lead to Difficulty

One of main takeaways of this chapter is that automation is not the only solution to address manual IT process workflows. There are times where it is less than optimal, or even harmful to introduce automation. There are also several ways that automation can be done very poorly, and result in a condition that is significantly worse than the condition that existed prior to the introduction of the automation. The considerations in this chapter can be used as a starting point in creating a robust, useful, and safe ITPA solution using the framework described in this paper. A final and important point is again one that Billing's makes that must be part of the architects' frame of mind when designing automation in the context of IT Process Automation.

"It is necessary that we look not only at the human or at the machines, but at the system, if we are co correct system faults or to design and implement more effective systems in the future." (Billings, 1997)

CONCLUSIONS AND FUTURE WORK

The Micro-Macro Orchestrator combination can be used in many ways to gain efficiencies. Particularly when implemented as described in this paper and a Publish-Subscribe messaging pattern. Publish-Subscribe as a management framework has only recently begun being used as an orchestration tool and is generally only utilized in one dimension or the other (ex: Micro (vertical) vs Macro (horizontal)). Combining both, as described in this paper, gives the ability to scale on a vertical and horizontal level using a single framework. This makes the proposed ITPA solution particularly useful in very large and complex IT service infrastructures. This paper describes a possible framework that can be used to perform any number of IT process and workflow automations, but the implementation details of the given workflows and process automations are left to the automation architect or system engineer. They will remain the final arbiter in determining the details around any specific automation or workflow.

The value in this approach can be found in the flexibility, speed, scale of execution, and instant feedback. These factors can lead to massive efficiencies, but can also lead to massive failures if not implemented with careful thought. For example just as one can properly update 10000 systems in parallel one can also improperly update or destroy those same 10000 systems in parallel and in an instant. This pushes the thoughts toward the safety of the system which was discussed somewhat in the last section of this paper, but not at an extent that it deserves. This topic in particular requires a significant amount of future work to understand the ramifications and potential implications of not being able to unscramble the egg, so to speak. However, this same concern would apply to any large scale ITPA, change management, provisioning, or IT management framework with a very large scope of execution.

The need for these types of solutions will only grow in the coming years as IT organizations throughout the world look to either expand their fleet services offered internally or outsource to a cloud or External IT Service provider. To either organization the solution described in this paper is applicable to gain the automation efficiencies necessary to scale out.

It should also be noted that the decoupled communications mechanism utilized in the proposed ITPA solution is also applicable in other areas that require the coordination and orchestration of numerous "systems." It doesn't seem like much of a stretch to see this type of orchestration mechanism used to orchestrate the lockstep movement of drones or other autonomous vehicles from a central controller with the ability to pass control to individual units for more complex movements and operations. The ability to both group and split tasks that are both vertically and horizontally complex is a highly relevant topic for any large scale computing environment and so this paper can serve as a gateway for further work.

REFERENCES

Creative Commons Attribution v3.0 . (2014, December 1). *STOMP Protocol Specification, V1.2.* Retrieved from GIT Hub: http://stomp.github.io/stomp-specification-1.2.html

Anonymous. (2013, November 2). *Idempotence*. Retrieved from Wikipedia: http://en.wikipedia.org/wiki/Idempotence

Billings, C. E. (1997). *The Search for a Human-Centered Approach (Human Factors in Transportation).* CRC Press.

Birman, K. P., & Joseph, T. A. (1987). *Exploiting Virtual Synchrony In Distributed Systems.* Ithaca: Cornell University, Department of Computer Science.

Colville, R. J. (2011). *IT Process Automation: Moving From Basics to Best Practices - REPORT - G00214344.* Stamford: Gartner Research.

Colville, R. J. (2011). *IT Process Automation: Run Book Automation Tools Mature to Broader Use.* Stamford: Gartner.

Garbani, J.-P., & O'Donnell, G. (2011). *Market Overview: IT Process Automation, Q3 2011.* Cambridge: Forrester Research.

Garbani, J.-P., Mendel, T., & Radcliffe, E. (2009). *The IT Automation Imperative - Putting IT On The Road to Industrial Mass Production REPORT -.* Cambridge MA: Forrester REsearch.

Kraveepetcharat, A. (2013). *Data Processing and Hosting Servcies in the US REPORT - 51821.* IBISWorld.

Leveson, N. G. (2011). *Engineering A Safer World - System Thinking Applied to Safety.* Cambridge: The MIT Press.

Moore, G. E. (1965). Cramming More Components Onto Integrated Circuits. *Electronics Magizine*, 4.

Natis, Y. V. (2012). *Gartner Reference Model for Elasticity and Multitenancy - REPORT - G00231615.* Stamford: Gartner.

Rance, Stuart; Hanna, Ashley; Hewlett-Packard. (2007, May 30). ITIL IT Service Management. *Glossary of Terms and Definitions*.

Uptime Institute, LLC. (2010-2013). *Data Center Site Infrastructure - Tier Standar: OPerational Sustainability.* New York: Uptime Institute.

Williams, D. (2010). *IT Operations Process Automation Achieves Mainstream Status With Increased Market Adoption and Importance - REPORT G00174293.* Stamford: Gartner.