The Information Technology (IT) Deployment Framework

****

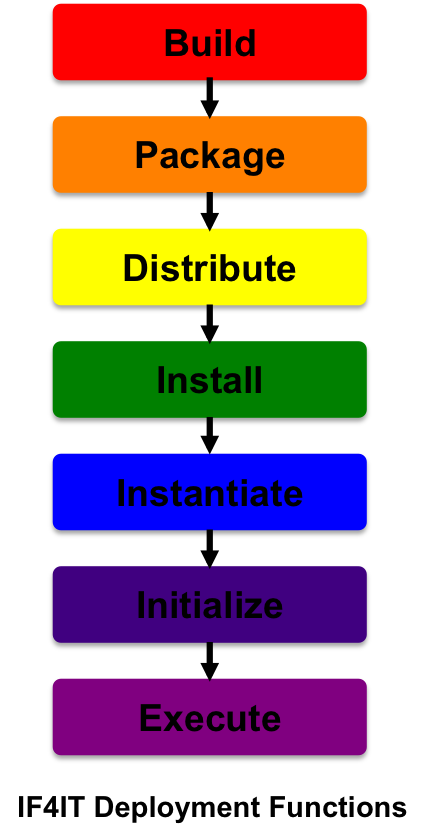
**LINK3 TECHNOLOGIES LTD.**

**Table of Contents**

Introduction to the IT Deployment Framework   
Understanding the Difference Between Deployment and Delivery   
The Benefits of Clearly Defining and Understanding Your Enterprise Deployment Framework   
Standard IT Deployment Functions   
Key Deployment Roles   
Key Deployment Organizations   
Metrics and Other Key IT Deployment Data   
Configuration Management for IT Deployment Framework Functions as a Best Practice   
Verification of Deployment Framework Functions as a Best Practice   
Automation of Deployment Framework Functions as a Best Practice   
How to Leverage the IT Deployment Framework for Better IT Project Plans and More Successful IT Delivery   
Important Summaries and Conclusions   
Learn More

**Abstract:**

*The IT Deployment Framework identifies, defines and describes the most fundamental functions that an IT Organization must perform, in each and every operating Environment, in order to successfully deliver one or more versioned Releases of IT Assets, such as Products, Software, Systems, and Applications to those targeted Environments for the purpose of execution by their intended End Users.*



## Introduction to the IT Deployment Framework

Before getting into the Deployment Framework, itself, we first have to understand the definition of Deployment.

A **Deployment**, which is the enterprise noun that is critical to the IT Discipline called Deployment Management, is defined as:

1. The set of repeatable, quantifiable and measurable activities that are performed within a specific operating Environment by IT Resources and Systems to successfully construct, deliver, and prepare a versioned IT Asset (e.g. Product, Software, System, Application) to that Environment for use by its targeted End Users.
2. The logistical act of delivering or releasing a Release to targeted End Users in a specific Environment.

To elaborate, these definitions focus on activities that are post-design and that are typically repeated, in each and every Environment to get such IT Assets up and running so that End Users can start to use them. It's important to know and understand each Environment has different targeted End Users. For example, a Training Environment has Trainers and Students as its End Users, while a User Acceptance Testing Environment has End Users that will only perform User Acceptance Tests. Each Environment has a different intended use and, therefore, rebuilding, re-delivering and re-preparing the Release in each Environment often requires some different things to occur for that Release to work appropriately within that specific Environment.

The activities or functions within a Deployment, which will be elaborated upon, further in this document, are trackable, measurable, and highly repeatable and most of these functions, if not all of them, can and should be automated, whenever possible, for best results. It is also important to understand that these activities occur within a given Environment and not across Environments or across phases of the SDLC.

The IT Deployment Framework is a decomposition of the most critical and fundamental functions that must be performed by IT Organizations, in each and every operating Environment, in order to successfully get enterprise Assets (e.g. Products, Software, Systems, etc.) up and running for relevant End Users.

The IT Deployment Framework is a subset or component of the greater Systems Development Life Cycle (SDLC) and is applicable in each of its operating environments.

Unlike in traditional Manufacturing industries, IT Organizations have the complexity of delivering Software and Software related Systems, such as Applications, which revolve around and must deal with the complexities associated with Software. This is not to dismiss the work associated with physical assets, such as Computing and Non-Computing Devices, however, it's always important to understand that IT Organizations exist to deliver and support Software. Software, unlike, tangible Assets that can fit into the traditional manufacturing mold, requires multiple iterations of construction, delivery, and testing, before it can ever be successfully delivered to its end users. It's these iterations that drive up the complexity and, if not done properly, that will also drive an enterprise into ruin. The IT Deployment Framework exists to facilitate and improve this complexity. It identifies, defines, and describes the key functions necessary to address the delivery of Software to each of its operating Environments.

*NOTE: The content within this document will continue to evolve, over time.*

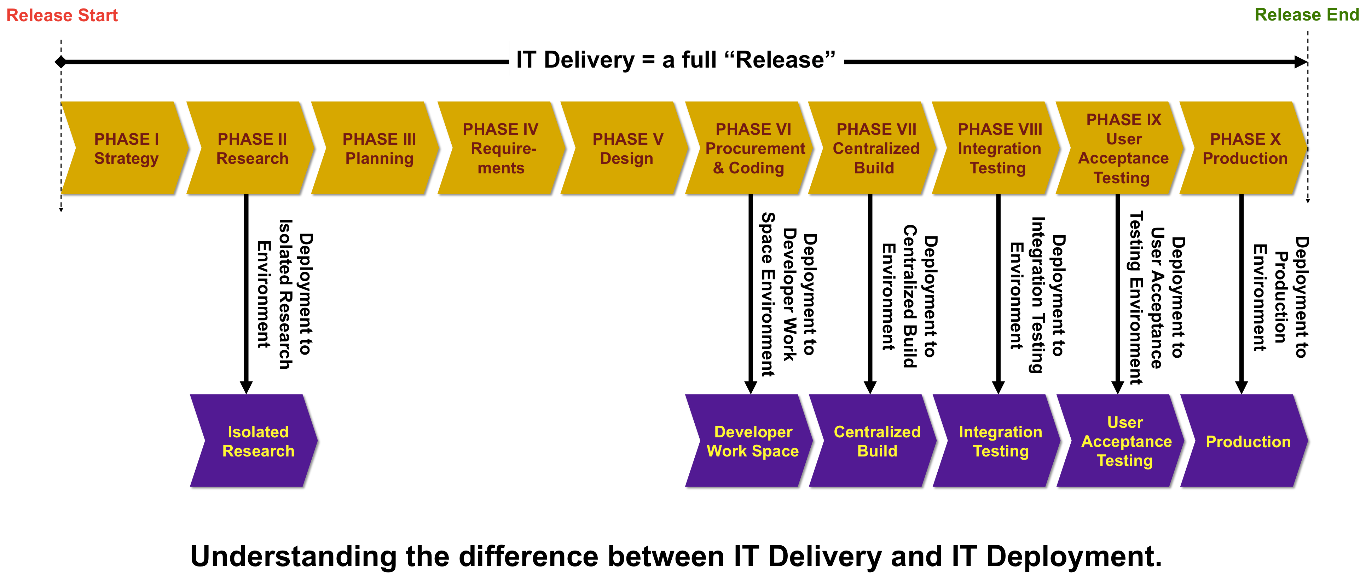
## Understanding the Difference between Deployment and Delivery/Rollout

It's absolutely critical for all IT Professionals to understand the differences between IT Delivery or IT Rollout, and IT Deployment so as to standardize as well as to eliminate any ambiguity associated with IT learning, conversations and communications.

In short, IT Delivery is equivalent to all work that is performed for an entire Release of Software or a System, across all Phases of the System Development Life Cycle (SDLC) (a.k.a. Software Development Life Cycle), from Release inception, through to complete closure.

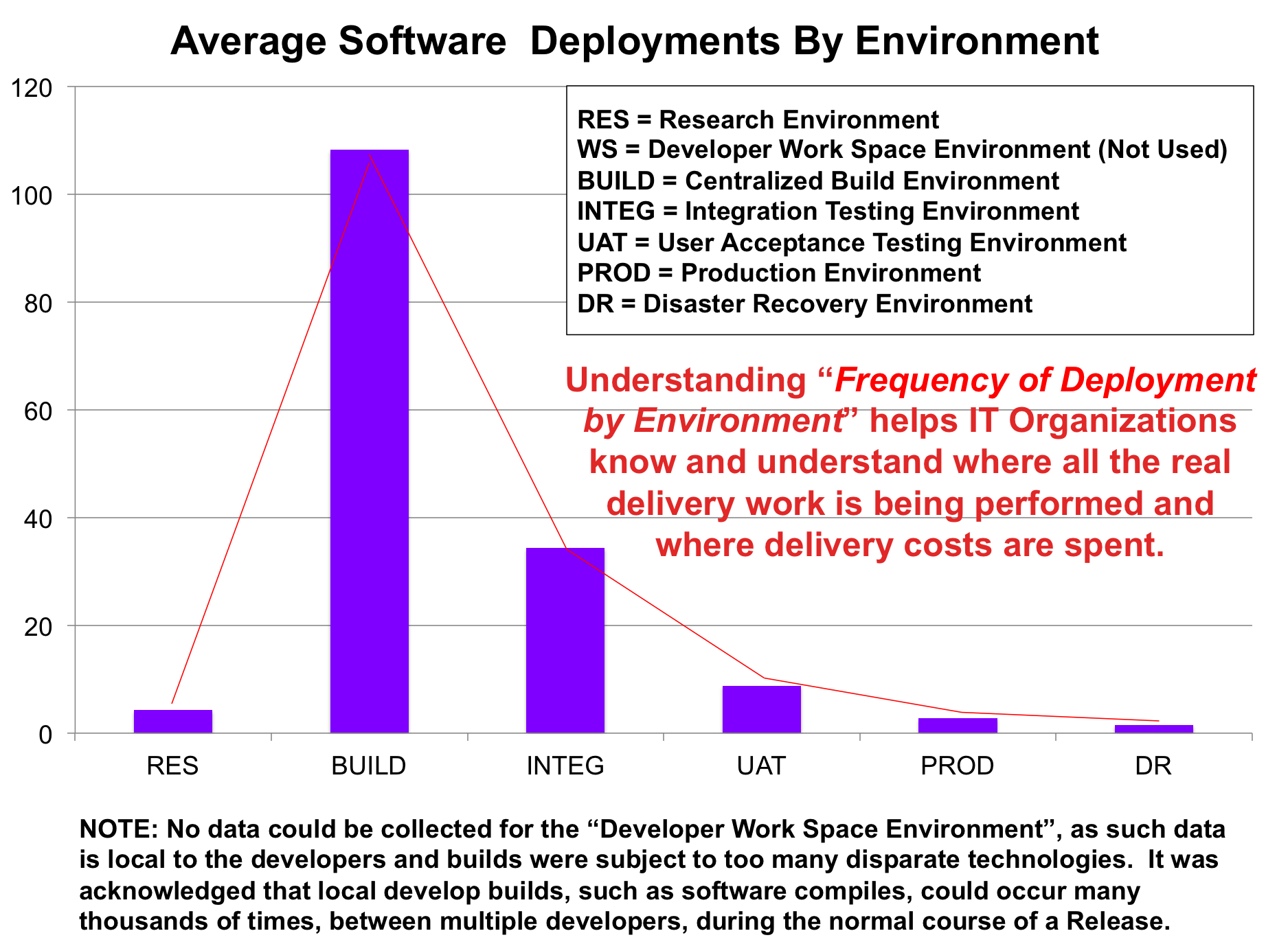
As mentioned earlier, IT Deployment represents the activities that are performed, over and over, throughout a Release to help prepare that Environment for its targeted End Users. So, for example, deploying to the User Acceptance Testing Environment implies preparing that Environment for the User Acceptance Testers, who are the End Users of that Operating Environment. And, deploying to the Production Environment implies preparing that Environment for the Production End Users.

The following diagram tries to help clarify the differences between IT Delivery and IT Deployment...



Delivery represents the end state and encompasses all of the work done throughout a Delivery Cycle, including Deployment, while Deployment represents all of the iterative and repetitive work done throughout a Delivery Cycle. It's important to understand that Deployment is performed, on average, more than once per targeted Environment. As a matter of fact, once your enterprise starts to measure its Deployments, it will find that most Deployment work is performed not in the Production Environment but in most of the Environments that precede Production.

While the actual frequency of Deployments by Environment may differ for different enterprises, the following graph, which was created by collecting data from a number of different enterprises in various industries, who proactively collected and measured their IT Deployments by Environment, shows an interesting pattern.



The graph highlights that, regardless of the industry, the bulk of IT Deployment work performed by enterprises who perform Software and Software-related Systems development is performed far more often in Environments like Centralized Build, Integration Testing and User Acceptance Testing than it is performed in Production and Disaster Recovery Environments. And, it's very important to note that while the graph represents an "averaging" of all contributor data and that data varied by enterprise and by industry, the real interesting piece of information is that each and every contributor had the exact same pattern for Deployment Frequency by Environment. As for why this happens, it should be clear that your IT Delivery teams spend most of their time trying to develop and stabilize their solutions, over and over again, before they ever get these solutions out to the "final" End Users in a Production Environment.

There are some very important conclusions an IT Professional can make from all of this...

* First, the bulk of an enterprise's IT Delivery work is "not" performed in its Production Environments but is actually performed in Environments that precede Production.
* Second, if an enterprise wants to improve its IT Delivery costs, timeframes, frequencies and quality, it makes sense that putting more effort into improving aspects of Delivery in earlier Operating Environments should yield better results, not only in downstream Environments, such as Production, but also across the entire SDLC.
* Third, because the bulk of all IT Delivery work is performed before deployment to an enterprise\s Production Environment, it makes sense that data created about the enterprise, throughout all earlier aspects of Deployment is fresher, more thorough and more easily collected while the work being performed rather than after the work has been performed. So, why, under any circumstance, would your enterprise focus larger efforts on trying to perform functions like Configuration Management "after" all that work and information is gone as opposed to fixing its Delivery and Deployment processes to help capture and maintain such information, proactively, throughout Delivery and Deployment?

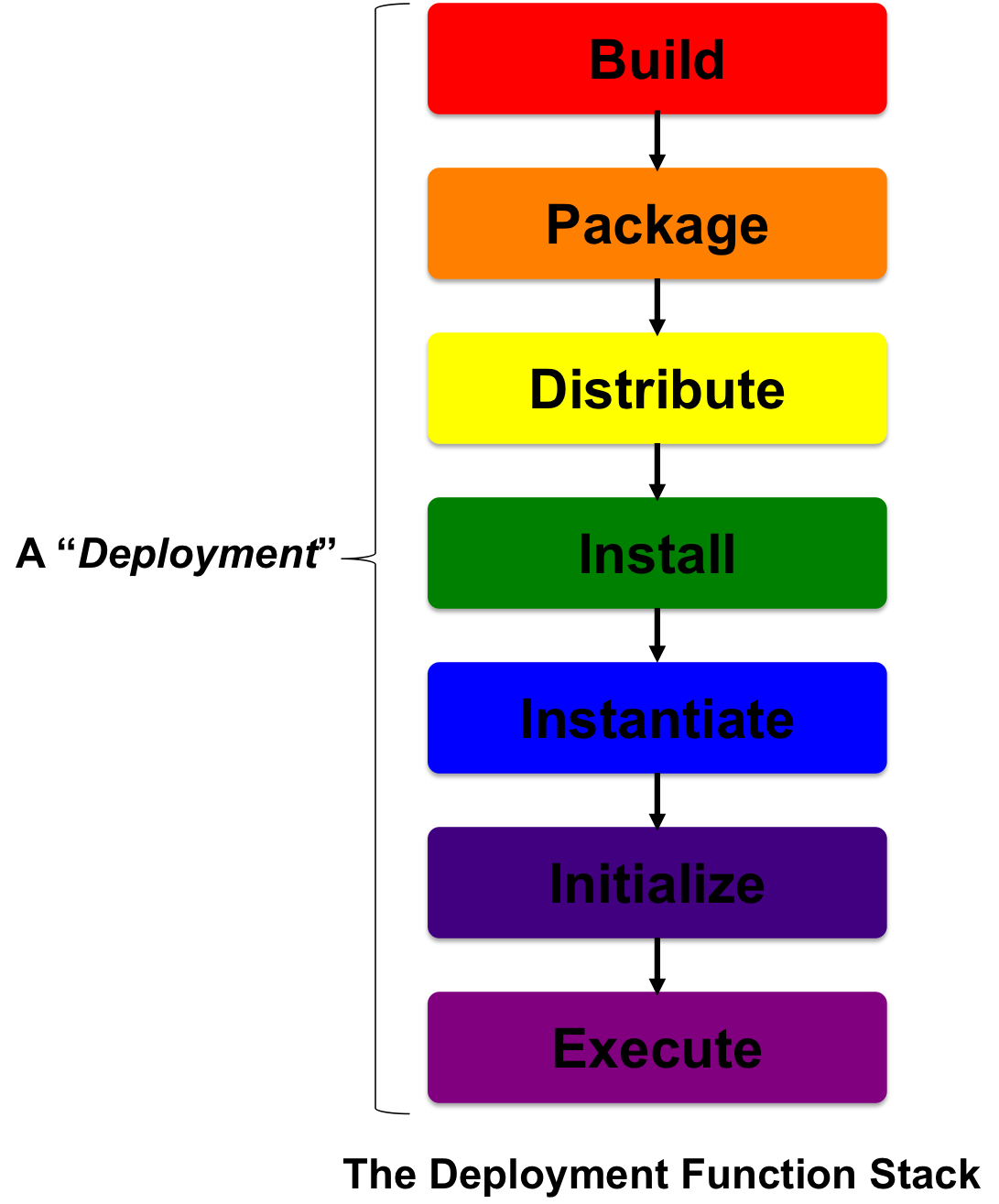
If you're a Reengineering or Six Sigma professional, or even if you're an IT manager or leader that just wants to improve IT performance, you may want to consider measuring your IT Deployment Frequencies by type of work performed and by targeted Deployment Environment. Doing so is one of the clearest forms of transparency into what IT organizations actually do for Delivery and provides tremendous insight into where money gets spent and why it gets spent. As for how to do so, this IT Deployment Framework will help you understand how to collect IT Deployment data and measure it.

## The Benefits of Clearly Defining and Understanding Your Enterprise Deployment Framework

There are multiple benefits for clearly identifying, defining, and describing the IT Deployment Framework for your enterprise. We believe that some of these benefits include but are not limited to:

* **Clarity:** Clear specification of work by those Resources performing the work. This helps eliminate ambiguity and makes things clear for all Roles.
* **Repeatability:** Repeatability is achieved only through clear definition of detailed work that highlights repeatable patterns. Clearly breaking down the functions associated with the IT Deployment Framework make such repeatability easier to achieve.
* **Rapid Delivery:** It's only after Deployment functions have been clearly identified, defined, assigned to specific roles, and made repeatable that an enterprise can do things like strive for automation in order to reduce their delivery times. Immature enterprises struggle because of manual deployment, whereas mature enterprises reap the benefits of automation
* **Quality:** Once Deployment functions have been clearly identified, defined, assigned to specific roles, and made repeatable, enterprises find that quality starts to go up, especially after the automation of such functions.
* **Reduced Costs:** Once Deployment functions have been clearly identified, defined, assigned to specific roles, and made repeatable, enterprises can clearly start the iterative process of driving the costs associated with each function, downward, especially after they understand how to apply automation to such functions.
* **Lower Delivery Risks:** Once Deployment functions have been clearly identified, defined, assigned to specific roles, and made repeatable, the risks associated with delivery of solutions starts to improve.
* **Better Perception of IT:** Only through the clear definition, repeatable execution, and successful delivery of IT Assets will the Businesses that fund IT view IT as a valuable partner for that Business' success.

## Standard IT Deployment Functions

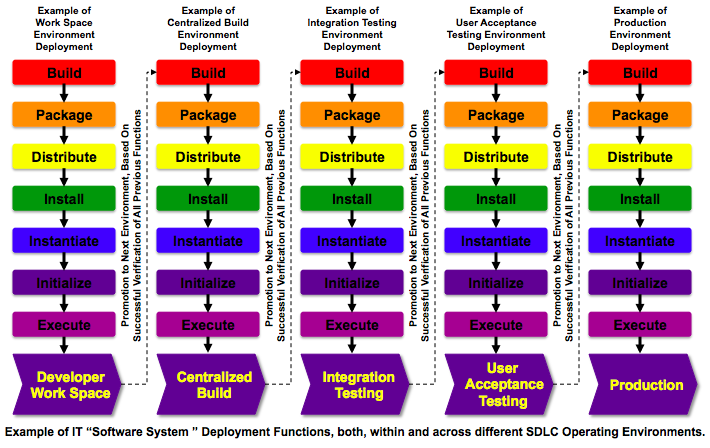


The standard functions that are required as part of any solid SDLC-based Deployment Framework include:

1. **Build:** (Definition of a [Build](http://www.if4it.com/SYNTHESIZED/GLOSSARY/B/Build.html); Parent IT Discipline is [Build Management](http://www.if4it.com/SYNTHESIZED/DISCIPLINES/Build_Management_Home_Page.html))
2. **Package:** (Definition of a [Package](http://www.if4it.com/SYNTHESIZED/GLOSSARY/P/Package.html); Parent IT Discipline is [Package Management](http://www.if4it.com/SYNTHESIZED/DISCIPLINES/Package_Management_Home_Page.html))
3. **Distribute:** (Definition of a [Distribution](http://www.if4it.com/SYNTHESIZED/GLOSSARY/D/Distribution.html); Parent IT Discipline is [Distribution Management](http://www.if4it.com/SYNTHESIZED/DISCIPLINES/Distribution_Management_Home_Page.html))
4. **Install:** (Definition of an [Installation](http://www.if4it.com/SYNTHESIZED/GLOSSARY/I/Installation.html); Parent IT Discipline is [Installation Management](http://www.if4it.com/SYNTHESIZED/DISCIPLINES/Installation_Management_Home_Page.html))
5. **Instantiate:** (Definition of an [Instantiation](http://www.if4it.com/SYNTHESIZED/GLOSSARY/I/Instantiation.html); Parent IT Discipline is [Instantiation Management](http://www.if4it.com/SYNTHESIZED/DISCIPLINES/Instantiation_Management_Home_Page.html))
6. **Initialize:** (Definition of an [Initialization](http://www.if4it.com/SYNTHESIZED/GLOSSARY/I/Initialization.html); Parent IT Discipline is [Initialization Management](http://www.if4it.com/SYNTHESIZED/DISCIPLINES/Initialization_Management_Home_Page.html))
7. **Execute:** (Definition of an [Execution](http://www.if4it.com/SYNTHESIZED/GLOSSARY/E/Execution.html); Parent IT Discipline is [Execution Management](http://www.if4it.com/SYNTHESIZED/DISCIPLINES/Execution_Management_Home_Page.html))

It's important to note that these functions are, more often than not, sequential and, therefore, become a dependency chain of work that must be performed to successfully Deploy any Asset to any specific operating Environment.

These functions are common to all SDLCs (whether they be Software Development Life Cycles or Systems Development Life Cycles) that exist to facilitate the development and delivery of software related solutions. This is true, regardless of the market segment or industry that an enterprise works in. The fact is that when you or your enterprise is delivering Software or Software related solutions that require some level of software coding or even scripting, these framework functions are and will most likely always be critical for the delivery process. It is important to note that the details of what happens for each function, in each unique operating Environment, is really constrained by the Environment, itself, and the intent of those delivering the solutions to their targeted Environment.



Having clear definition and control of all IT Deployment Functions, both, within and across each environment is the key to transparency that is both deep and wide, with correlation between the things you're looking at. It's also one of the most important ingredients to IT success. In companies that are heavy on manufacturing, many Industrial Engineers are hired to not only improve the products that are delivered but also to consistently improve "*the plant*," which includes everything from the assembly lines, to roles and responsibilities, to the facilities. Like those enterprises that are constantly using Industrial Engineering as a means to iteratively improve everything, IT Organizations should constantly be working to improve their own Deployment processes. This starts with knowing, understanding, and proactively managing all of your enterprise's critical IT Deployment Functions.

## Key Deployment Roles

The following covers the most common Roles associated with the IT Deployment Framework.

*Note about the following: "ASSET\_TYPE" can be replaced with whatever appropriate Asset it is that is applicable, such as Software, or Database, etc.*

**Worker Roles: The Resources or Systems that perform the actual Deployment work.**

|  |  |  |
| --- | --- | --- |
| "Deployer" or "[ASSET\_TYPE] Deployer" | IT Role IT Role | 1. The Resource that is responsible for initiating and overseeing the Deployment of one or more Assets, from start to finish. 2. The System or Automation Framework that is responsible for the Deployment process. |
| "Builder" or "[ASSET\_TYPE] Builder" | IT Role IT Role | 1. The Resource that is responsible for initiating and overseeing the complete Build of one or more Assets, from start to finish. 2. The System or Automation Framework that is responsible for the complete Build process. |
| "Packager" or "[ASSET\_TYPE] Packager" | IT Role IT Role | 1. The Resource that is responsible for initiating and overseeing the complete Packaging process of one or more Assets, from start to finish. 2. The System or Automation Framework that is responsible for the complete Packaging process. |
| "Distributor" or "[ASSET\_TYPE] Distributor" | IT Role IT Role | 1. The Resource that is responsible for initiating and overseeing the complete Distribution process of one or more Assets, from start to finish. 2. The System or Automation Framework that is responsible for the complete Distribution process. |
| "Installer" or "[ASSET\_TYPE] Installer" | IT Role IT Role | 1. The Resource that is responsible for initiating and overseeing the complete Installation process of one or more Assets, from start to finish. 2. The System or Automation Framework that is responsible for the complete Installation process. |
| "Instantiator" or "[ASSET\_TYPE] Instantiator" | IT Role IT Role | 1. The Resource that is responsible for initiating and overseeing the complete Instantiation process of one or more Assets, from start to finish. 2. The System or Automation Framework that is responsible for the complete Instantiation process. |
| "Initializer" or "[ASSET\_TYPE] Initializer" | IT Role IT Role | 1. The Resource that is responsible for initiating and overseeing the complete Initialization process of one or more Assets, from start to finish. 2. The System or Automation Framework that is responsible for the complete Initialization process. |
| "Executor" or "[ASSET\_TYPE] Executor" | IT Role IT Role | 1. The Resource that is responsible for initiating and overseeing the complete Execution process of one or more Assets, from start to finish. 2. The System or Automation Framework that is responsible for the complete Execution process. |

**Ownership Roles: The Resources that own and are accountable for the actual Deployment work.**

|  |  |  |
| --- | --- | --- |
| "Deployment Owner" or "[ASSET\_TYPE] Deployment Owner" | IT Role | The Resource that owns and is accountable for the Deployment of one or more Assets, from start to finish. |
| "Build Owner" or "[ASSET\_TYPE] Build Owner" | IT Role | The Resource that owns and is accountable for the Build of one or more Assets, from start to finish. |
| "Packaging Owner" or "[ASSET\_TYPE] Packaging Owner" | IT Role | The Resource that owns and is accountable for the Packaging of one or more Assets, from start to finish. |
| "Distribution Owner" or "[ASSET\_TYPE] Distribution Owner" | IT Role | The Resource that owns and is accountable for the Distribution of one or more Assets, from start to finish. |
| "Installation Owner" or "[ASSET\_TYPE] Installation Owner" | IT Role | The Resource that owns and is accountable for the Installation of one or more Assets, from start to finish. |
| "Instantiation Owner" or "[ASSET\_TYPE] Instantiation Owner" | IT Role | The Resource that owns and is accountable for the Instantiation of one or more Assets, from start to finish. |
| "Initialization Owner" or "[ASSET\_TYPE] Initialization Owner" | IT Role | The Resource that owns and is accountable for the Initialization of one or more Assets, from start to finish. |
| "Execution Owner" or "[ASSET\_TYPE] Execution Owner" | IT Role | The Resource that owns and is accountable for the Execution of one or more Assets, from start to finish. |

**Management Roles: The Resources that manage Deployment related work.**

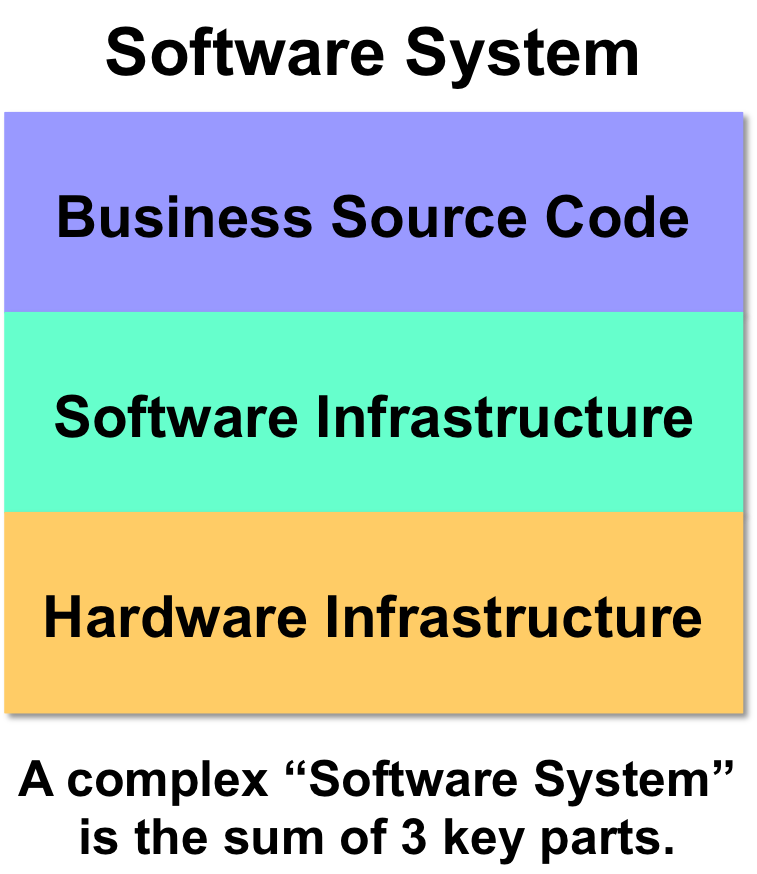
|  |  |  |
| --- | --- | --- |
| "Deployment Manager" or "[ASSET\_TYPE] Deployment Manager" | IT Role | The Resource that manages the Deployment process (or one or more pieces of it), on behalf of or as a delegate of its Owner. |
| "Build Manager" or "[ASSET\_TYPE] Build Manager" | IT Role | The Resource that manages the Build process (or one or more pieces of it), on behalf of or as a delegate of its Owner. |
| "Packaging Manager" or "[ASSET\_TYPE] Packaging Manager" | IT Role | The Resource that manages the Packaging process (or one or more pieces of it), on behalf of or as a delegate of its Owner. |
| "Distribution Manager" or "[ASSET\_TYPE] Distribution Manager" | IT Role | The Resource that manages the Distribution process (or one or more pieces of it), on behalf of or as a delegate of its Owner. |
| "Installation Manager" or "[ASSET\_TYPE] Installation Manager" | IT Role | The Resource that manages the Installation process (or one or more pieces of it), on behalf of or as a delegate of its Owner. |
| "Instantiation Manager" or "[ASSET\_TYPE] Instantiation Manager" | IT Role | The Resource that manages the Instantiation process (or one or more pieces of it), on behalf of or as a delegate of its Owner. |
| "Initialization Manager" or "[ASSET\_TYPE] Initialization Manager" | IT Role | The Resource that manages the Initialization process (or one or more pieces of it), on behalf of or as a delegate of its Owner. |
| "Execution Manager" or "[ASSET\_TYPE] Execution Manager" | IT Role | The Resource that manages the Execution process (or one or more pieces of it), on behalf of or as a delegate of its Owner. |

**Important Note About Roles:** The use of the term *"Roles"* does not automatically imply different people or different systems. In fact, in small enterprises, one person may perform all roles, whereas, in lager enterprises, each role may be broken out across separate people. In smart enterprises, regardless of their size, they've automated each role and are executing each through the use of one common *automation framework.*

## Key Deployment Organizations

In order to fully understand the Organizations that perform IT Deployment Functions, it's important to first clarify the key Roles that leverage such functions. The entire Deployment Framework revolves around "Software" and the Systems that exist to support Software. Traditionally, the term "Software Development" has always been a bit overused. Most people use the term in a very vague and broad manner but rarely break out the details of what it means. As we go further into the topic of Deployment Organizations, we'll find that the scope of what a "Software Developer" is will become less broad, this will become especially evident as the size of the enterprise type we're discussing grows from a small enterprise to a large enterprise.

For the purpose of this topic, we will need to break out three relevant areas of IT that will drive how Organizations are structured, especially as they grow. These three areas are Business Software Development, Software Engineering, and Hardware Engineering.



* **Business Software Development:** Focuses on the "business rules" that are part of the actual business purpose of the solution being developed, specifically for its targeted End Users. This is often referred to as Business Application Development or Software Development.
* **Software Engineering:** Focuses on all of the Software Infrastructure that the Business Application will leverage. This includes but is not limited to examples such as Databases, Batch Infrastructure, Application Monitoring Infrastructure, Extract, Transformation and Load (ETL) Infrastructure, etc. In short and with the exception of things like Databases, which are common to most Business Applications these days, such Software Infrastructure are considered commodity solutions that mid-to-larger sized enterprises acquire from vendors or leverage from Open Source communities. Doing so ensures that the Business Software Developer does not have to build such complex Software Infrastructure, him or herself, from scratch. The intent is to is to acknowledge that Software Developers cannot be experts in everything and, therefore, to allow Business Application Developers to focus more on the "value add" aspect of Business Rules that more closely apply to the needs of their End Users and less on the complex Software Infrastructure that can be acquired from experts who pour large investments into such constructs.
* **Hardware Engineering:** Focuses on things that include but are not limited to computing devices, non-computing devices, storage, network, etc. When building small Software Applications in small enterprises, letting developers control and manage their own Hardware Infrastructure is pretty normal. However, as Software Applications grow more complex, it becomes more realistic that such functions are broken out, into separate organizations that have greater expertise on such matters. Like Software Infrastructure, the point is that there are experts that know more about these things, leaving Software Developers to focus more on the "business value add" areas of their work.

The point to take away from the above three Roles is that a complex Software System or Business System is composed of all three types of solutions that are interdependent on each other. Therefore, all three have a hand in IT Deployment and how each is handled changes with the size of an enterprise, as we'll see further on. As a matter of fact, experience tells us that the more complex a System is, the larger the Software Engineering and Infrastructure portion of the System is, making Deployment less about the Business Software and its related Source Code and more about all of the Software Infrastructure that such Source Code depends on.



**The Small Enterprise:** In a small enterprise, Software Developers perform all Software related functions. What this means is that they not only perform Design, Coding, Test and Deployment but they also perform and are accountable for Software Engineering, and Hardware Engineering responsibilities. In small enterprises, these engineering functions are "mixed in" with the responsibilities of Software Designers and Coders.

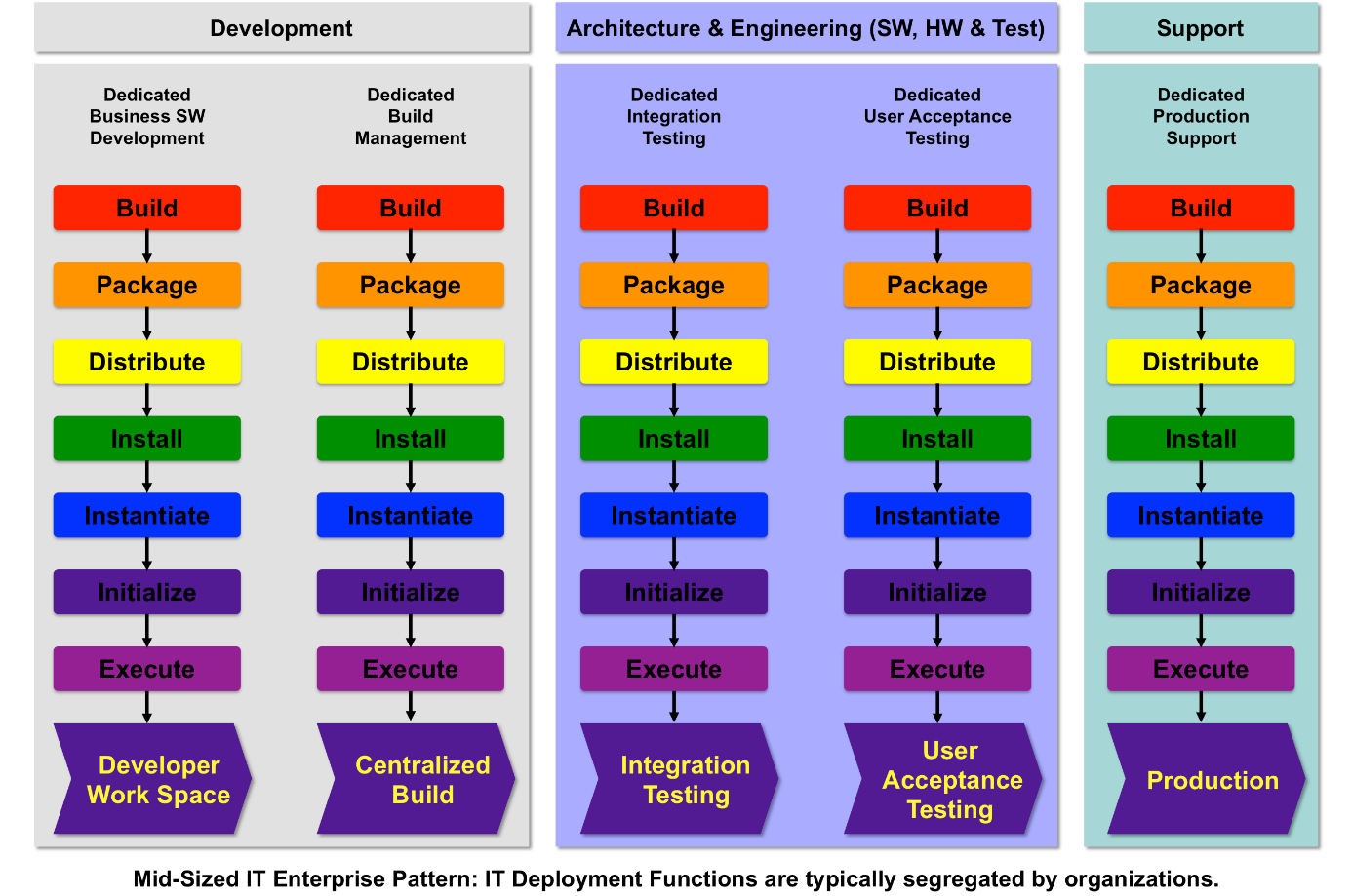
The structure of IT Organizations that perform IT Deployment functions often varies with the size of the enterprise. Small enterprises that have local footprints, where everyone is located in one common location behave far differently than very large enterprises that are global and geographically dispersed. It's very important to realize that one model does not fit all needs for all enterprises.

For example, in a very small enterprise, all IT Deployment Functions are often performed by the very same people, who often all exist in one common Software Development Team. This team would perform all functions in all environments. (*NOTE: A smart small enterprise still highly defines and separates its environments, even if done logically and not physically.*)



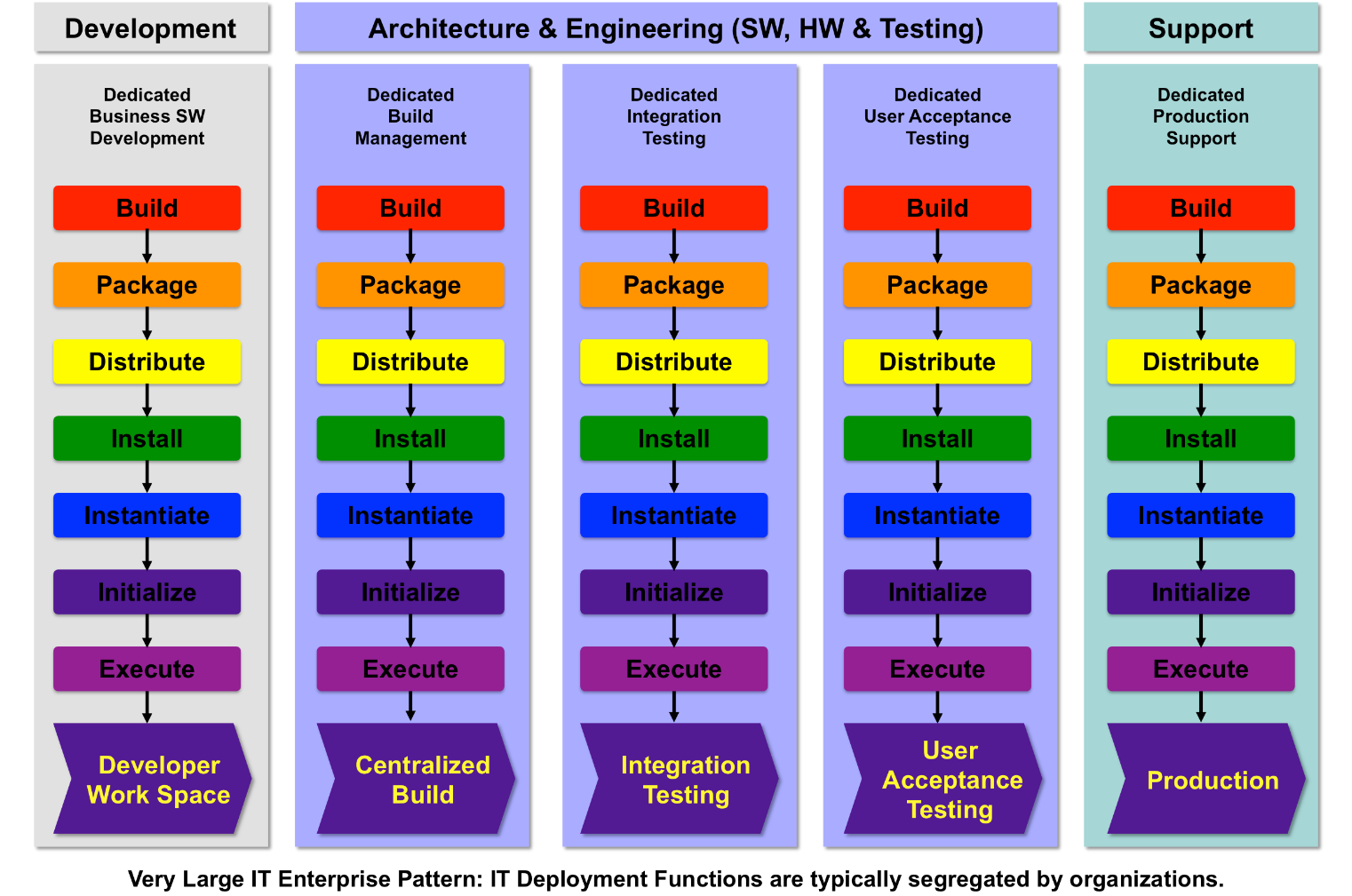
**The Mid-Sized Enterprise:** As enterprises grow, it becomes more difficult to have one group perform all functions and leaders will start to split functions into different Organizations, hopefully, with the intent to better handle and improve scaling work. Most often, the splitting occurs in a manner that starts to lock developers and engineers out of the Production Operating Environment.

The common pattern is to start to separate Development and Support functions from the types of things that are typically considered to be more closely related to Architecture, Engineering and Testing. As a result, Software Engineering and Infrastructure, as well as Hardware Engineering and Infrastructure, start to become clearer areas of IT that require more investment and attention. Centralization and reuse of Software Infrastructure and Hardware Infrastructure start to become goals to reduce costs and drive up consistency. This starts leading to IT Shared Services or Centers of Excellence that focus on these types of things, so that Business Application Developers can focus more closely on business facing solutions. Also, some businesses will now start to attempt some outsourcing of areas such as Software Development (the left-most side of the diagram) and Support (the right-most side of the diagram).



**The Large Enterprise:** When an enterprise grows to have very large IT (companies that have many tens and even hundreds of thousands of employees are the clearest examples because of their sizes), IT Organizations are typically split up into one of many different permutations to attempt to handle large IT volumes for a large business enterprise whose stakeholders are geographically dispersed. The structuring of these very large IT organizations varies but you'll notice from the following visual pattern that a common approach is to decouple Software Development and Support from everything else, which aligns to many large enterprise's outsourcing models. Done correctly, you can outsource Development and Support but almost all enterprises learn and come to the conclusion (sometimes very painfully) that you can almost never outsource Architecture, Engineering and all of the core IT Deployment functions that are performed while gluing everything together. Such functions represent the heart and purpose of an IT Organization, which is getting things delivered and working successfully.

As these patterns evolve in larger enterprises, the need for Software Engineering and Hardware engineering organizations become clearer and clearer.



Having an IT Deployment Framework becomes even more critical in the cases where Business Application or Software Development is outsourced. In such cases, if an enterprise doesn't have a clear Framework for taking in and integrating Software that was developed by a third party, chances are that Organization in dealing with chaos.

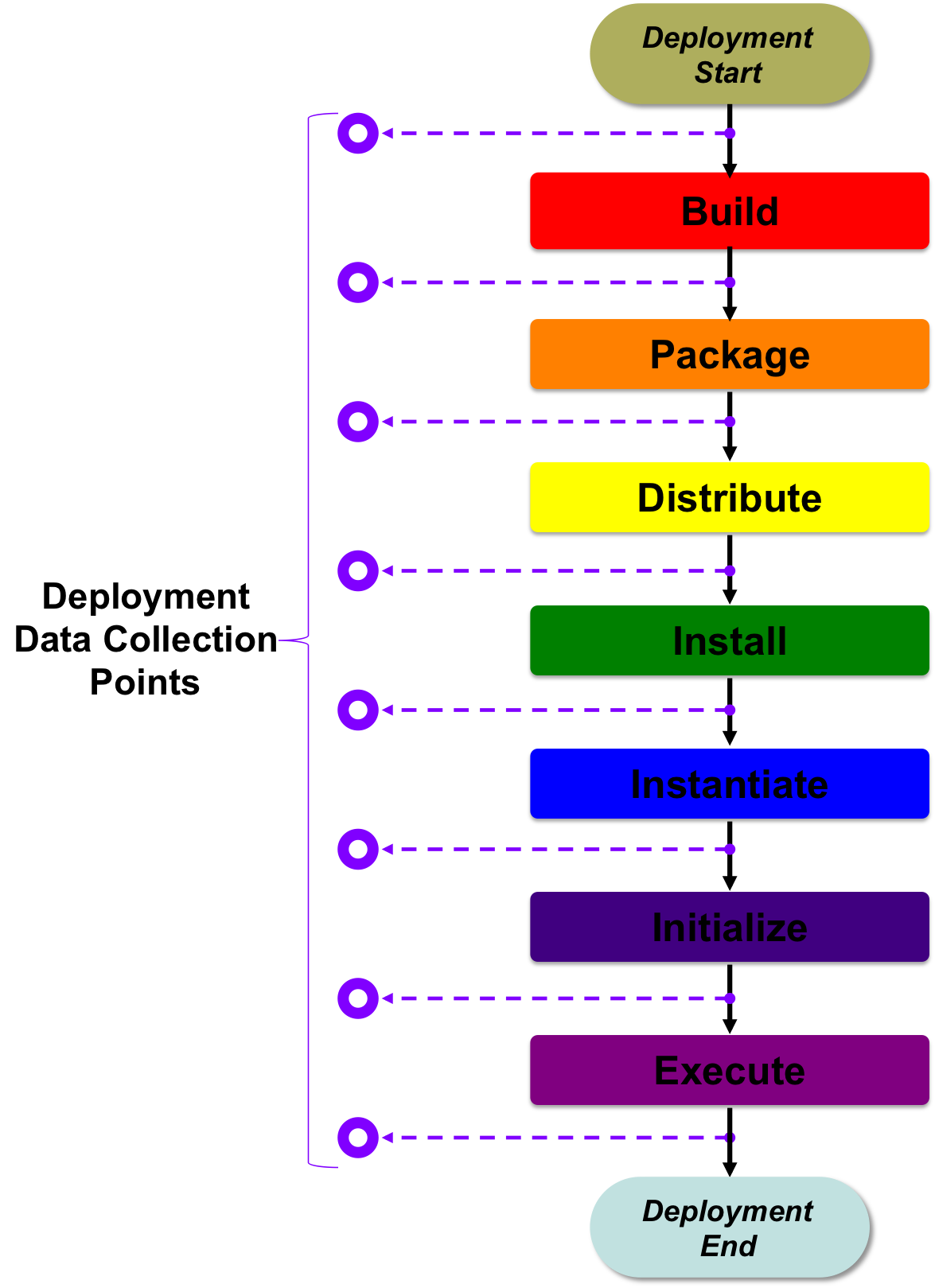
The only Foundation recommended key Organization for the purposes of owning, managing, and executing deployment functions is the IT Deployment Organization (often labeled or housed within Engineering). If necessary, this Organization can be broken into two separate Organizations, where one deals wholly with Software deployment (Software Engineering) and another deals wholly with physical infrastructure deployment (Hardware Engineering) of all forms.

If the Deployment Organization grows to be very large, then the Owner of the Deployment Organization can consider partitioning this Organization into sub-Organizations that align with the related Deployment Functions.

A note on the complexity of Deployment... If you think about it, done correctly, your Deployment Organization will know how to build everything, know where it's all built, know where everything is, and know how it all ties together and works. It's for this reason that you may want to consider staffing this Organization with full time Resources that represent some of your most talented Software and Hardware Engineers, as well as consider making this Organization the constant that you don't outsource. This is especially true if you give them the responsibility for automating all Deployment Functions. Such a Deployment Organization typically acts like your air traffic controller, coordinating all work that is going on for all Assets in all operating Environments.

## Metrics and Other Key IT Deployment Data

Before we can even discuss the collection of key data for Metrics we have to understand where data can be collected, consistently, from each of the different touch-points in the IT Deployment Framework. The framework is set up to ensure consistent patterns of work in each Operating Environment (Dev, Integ, UAT, Prod, etc.). This consistency ensures that the touch-points for data collection will be the same for each and every IT Deployment Function (Build, Package, etc.), regardless of the Operating Environment that function is being executed in, because the same functions are executed relative to their same position, in each Environment. As a result, data can be consistently collected before and after every IT Deployment Function, in each and every environment that these functions are executed in.



At a minimum, the start and end of each function can be registered. For advanced organizations that have automated their IT Deployment Functions, the execution of each function becomes a rich information collection platform. Done correctly, examples of the type of data that can be collected includes:

* **Who:** Who ran or executed the function (i.e. the Build, the Packaging, the Distribution, etc.).
* **What:** Exactly what function was executed for what specific Release of what specific Asset (Product, Software, System, etc.) and what other objects are involved in or are being used as part of functional execution, such as key Configuration Items (e.g. Computers, Software Constructs, Configuration Data, etc.).
* **When:** The start time, the end time, and any key check-point times that are in between, given that automation can communicate start and stop times for any sub-steps that are being executed as part of any IT Deployment Function.
* **Where:** Where the IT Deployment Function is being executed, such as the Operating Environment in which the function is being executed or the facility in which the function is being executed (that is, if you've been smart about how your systems are tied to your facilities).
* **How:** How things are connected together, through relationships that are, both, dependencies for and the outcome of each IT Deployment Function (*All of you dedicated Configuration Managers may want to pay very close and special attention to this!*).
* **Why:** Why things didn't pass if a function stops before successful execution.

Once an IT Organization begins to standardize the execution of its IT Deployment Functions it can start to collect vast amounts of truly relevant operational data that is critical to all aspects of IT Delivery. You and your enterprise can start to collect *Build, Packaging, Distribution, Installation, Instantiation, Initialization and Execution* data about each and every Release of each and every Asset that is important to you, in each and every Environment. It is only after the rigorous collection and constant analysis of such data that an enterprise can truly understand and begin to improve its delivery performance and quality, while lowering its costs.

Most Organizations that follow these practices become transparent and start to use that transparency to continuously improve, specifically with the intent to drive delivery time down, drive quality up, drive risk down, and drive costs down. Organizations that do not follow such practices live in a state of constant chaos, no real transparency, wondering why things fail, and never improving. Think about it... If you want your IT Organization to have high marks and a perception of satisfaction with its customers, rigid repeatability and high levels of transparency are the primary ingredients to success. There is no greater area to achieve quick impact and gain such successes than starting with the improvement of your IT Deployment processes. If your enterprise is littered with expensive systems and software, and you don't know "exactly" what you own, where it is, how its being used, and who is using it, and what will be impacted if any of it changes, chances are that your enterprise did not use a rigid IT Deployment Framework to achieve such levels of governance and now you're stuck with cleaning up the mess.

**About Metrics:** If you haven't figured it out, yet, the collection of metrics comes from rigid repeatability, in this case of each IT Deployment Function. Once you've made your IT Deployment processes and functions highly repeatable, you can start to collect...

*Time Based Metrics Like:*

* Time to Build (in each Environment for each Release of every Asset (Software, Systems, etc.)
* Time to Package (in each Environment for each Release of every Asset (Software, Systems, etc.)
* Time to Distribute (in each Environment for each Release of every Asset (Software, Systems, etc.)
* Time to Install (in each Environment for each Release of every Asset (Software, Systems, etc.)
* Time to Instantiate (in each Environment for each Release of every Asset (Software, Systems, etc.)
* Time to Initialize (in each Environment for each Release of every Asset (Software, Systems, etc.)
* Time to Execute (in each Environment for each Release of every Asset (Software, Systems, etc.)

*Quantity Based Metrics Like:*

* The number of Builds, Packages, Distributions, Installations, Instantiations, Initializations and Executions per any period for any and all Releases of all Assets in all Environments.
* The number of Configuration Items involved in or used for the execution of each IT Deployment function, for each and every Release, of each and every Asset, in each and every Environment.
* The number of Resources involved in the execution of each and every IT Deployment Function, for each and every Release, of each and every Asset, in each and every Environment.

*Quality Based Metrics Like:*

* Build Failures and Successes as compared to other Asset Releases for the same Asset or even when measured against other Assets, in each and every Operating Environment.
* Packaging Failures and Successes as compared to other Asset Releases for the same Asset or even when measured against other Assets, in each and every Operating Environment.
* Distribution Failures and Successes as compared to other Asset Releases for the same Asset or even when measured against other Assets, in each and every Operating Environment.
* Installation Failures and Successes as compared to other Asset Releases for the same Asset or even when measured against other Assets, in each and every Operating Environment.
* Instantiation Failures and Successes as compared to other Asset Releases for the same Asset or even when measured against other Assets, in each and every Operating Environment.
* Initialization Failures and Successes as compared to other Asset Releases for the same Asset or even when measured against other Assets, in each and every Operating Environment.
* Execution Failures and Successes as compared to other Asset Releases for the same Asset or even when measured against other Assets.

**For Six Sigma and Other Quality Improvement Experts:** If you're a Six Sigma expert or some other certified improvement specialist, you may want to take a close look at how your IT Organization performs its Deployment Functions for all Releases of all Assets. While IT Deployment is certainly not the easiest thing to fix, if it's broken, working to improve it certainly has very tangible and extremely visible results. Also, think about pushing "automation" as a solution for achieving operational gains. Business Process Management (BPM) is not just for your business clients but also for how your IT Organization runs its own business of IT.

## Configuration Management for IT Deployment Framework Functions as a Best Practice

Each function in the IT Deployment Framework provides critical data about an IT Asset, as is related to that function. When combined, the Configuration Management data that is collected from all of the functions, in each of the Environments, yields one of the most complete Configuration Management profiles that can be achieved in IT, about any specific asset.

In short, if your enterprise automates its Deployments by automating as many of its Deployment Functions as possible, and your enterprise does this for all or most of its applications, the automation tools will collect critical Configuration Management and profile data about that asset for each and every Deployment in each and every environment.

Deployment Configuration Management is the sum of all Configuration Management that is performed for each IT Deployment Function. This list includes:

* Build Configuration Management : yields Build specific Configuration Data for a specific version of a specific Asset or set of Assets in a specific environment by specific people and roles.
* Package Configuration Management : yields Package or Packaging specific Configuration Data for a specific version of a specific Asset or set of Assets in a specific environment by specific people and roles.
* Distribution Configuration Management : yields Distribution specific Configuration Data for a specific version of a specific Asset or set of Assets in a specific environment by specific people and roles.
* Installation Configuration Management : yields Installation specific Configuration Data for a specific version of a specific Asset or set of Assets in a specific environment by specific people and roles.
* Instantiation Configuration Management : yields Instantiation specific Configuration Data for a specific version of a specific Asset or set of Assets in a specific environment by specific people and roles.
* Initialization Configuration Management : yields Initialization specific Configuration Data for a specific version of a specific Asset or set of Assets in a specific environment by specific people and roles.
* Execution Configuration Management : yields Execution specific Configuration Data for a specific version of a specific Asset or set of Assets in a specific environment by specific people and roles.

If you decompose each IT Deployment Function, you can see that each function has a Configuration Dependency on the previous Function in the function stack:

* **Execution** *depends on successful* **Initialization**
* **Initialization** *depends on successful* **Instantiation**
* **Instantiation** *depends on successful* **Installation**
* **Installation** *depends on successful* **Distribution**
* **Distribution** *depends on successful* **Packaging**
* **Packaging** *depends on successful* **Building**

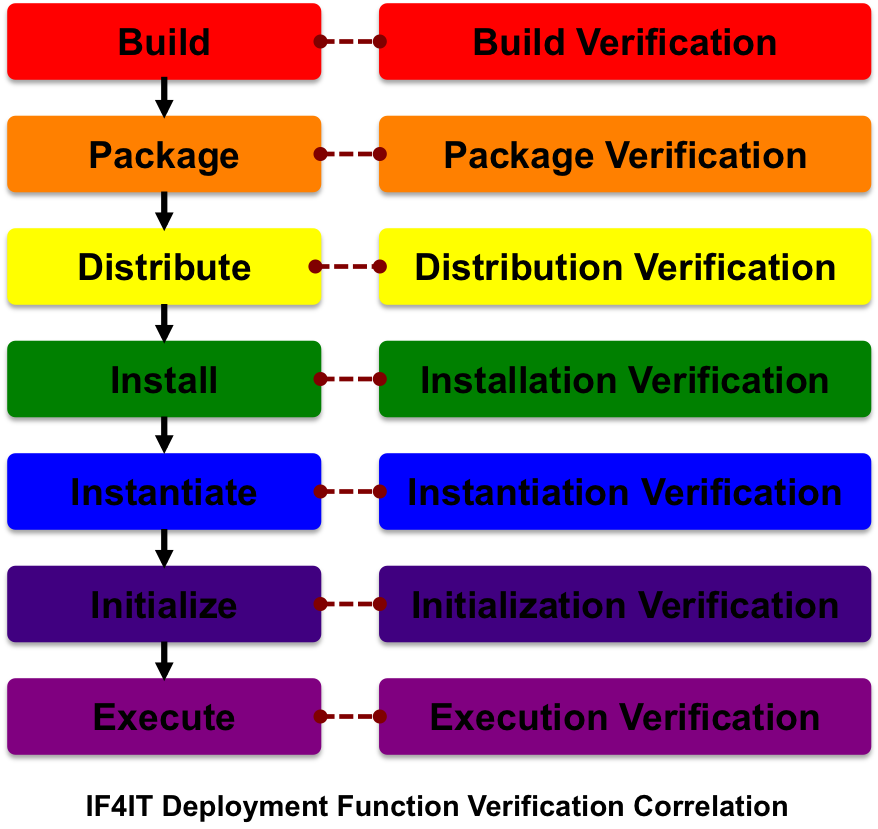
What this all leads to is that if and IT Organization works to collect all Configuration Data from all IT Deployment Functions for all Assets that are deployed through such a framework, that Organization will have a truly powerful 360 degree view of everything tied to a specific version of a specific Asset (i.e. a Release) in a specific environment. Collecting such data and properly putting it into constructs like Semantic Data Repositories (e.g. RDF based) will allow common Configuration Items to bridge information across the details of different Assets that have been built. However, this topic revolves around Enterprise Information Management and is a whole other level of complexity that is not intended to be solved as part of this framework. One final note on this is that such Semantic Repositories are typically far more useful than so called Configuration Management Databases (CMDBs) that do not implement Semantic Data Schemas or Models because they show semantic linkages or relationships between any two nodes (things) that are tied together through any specified relationship that would make sense for such a connection.

One of the biggest recipes for failed Configuration Management efforts can be attributed to enterprises trying to manually capture and maintain Configuration Data after (in most cases long after) Assets have been deployed to their operating Environments. While the Foundation understands that a leader inheriting an enterprise that doesn't know and understand its current state is one problem, it's a completely different problem if that leader does not acknowledge and implement real-time Deployment Configuration Management solutions to enable the proactive capturing and management of Configuration Management Data, automatically, as soon as it's generated by the function that is being executed.



## Verification of Deployment Framework Functions as a Best Practice

Each Deployment Function has associated verification that must be performed to ensure that the function was executed properly. For example, rules must be put in place to define what it means to successfully complete the *Build* portion of a Deployment into a Centralized Build Environment, vs. an Integration Testing Environment, vs. a User Acceptance Testing Environment, etc. The same holds for all IT Deployment Functions. The quality of delivery is tied not only to the functionality of the Asset being delivered but also to the processes that help deliver that Asset for use to its targeted environment. Deployment function verification correlation follows:



It is considered a best practice to verify (a.k.a. "Test") the successful completion of each and every IT Deployment Function in each and every targeted operational Environment. In most cases, such verification should be automated, as should the IT Deployment Functions, themselves.

A note on Execution Verification: Verification of the IT Deployment Function called "Execution" means different things for different environments. For example, in the *Centralized Build Environment* Execution is successful after the execution of all Unit and Module Tests, whereas in your *User Acceptance Testing Environment* successful execution often means the successful execution of all Unit and Module Tests, all Integration Tests, and all User Acceptance Tests.

## Automation of Deployment Framework Functions as a Best Practice

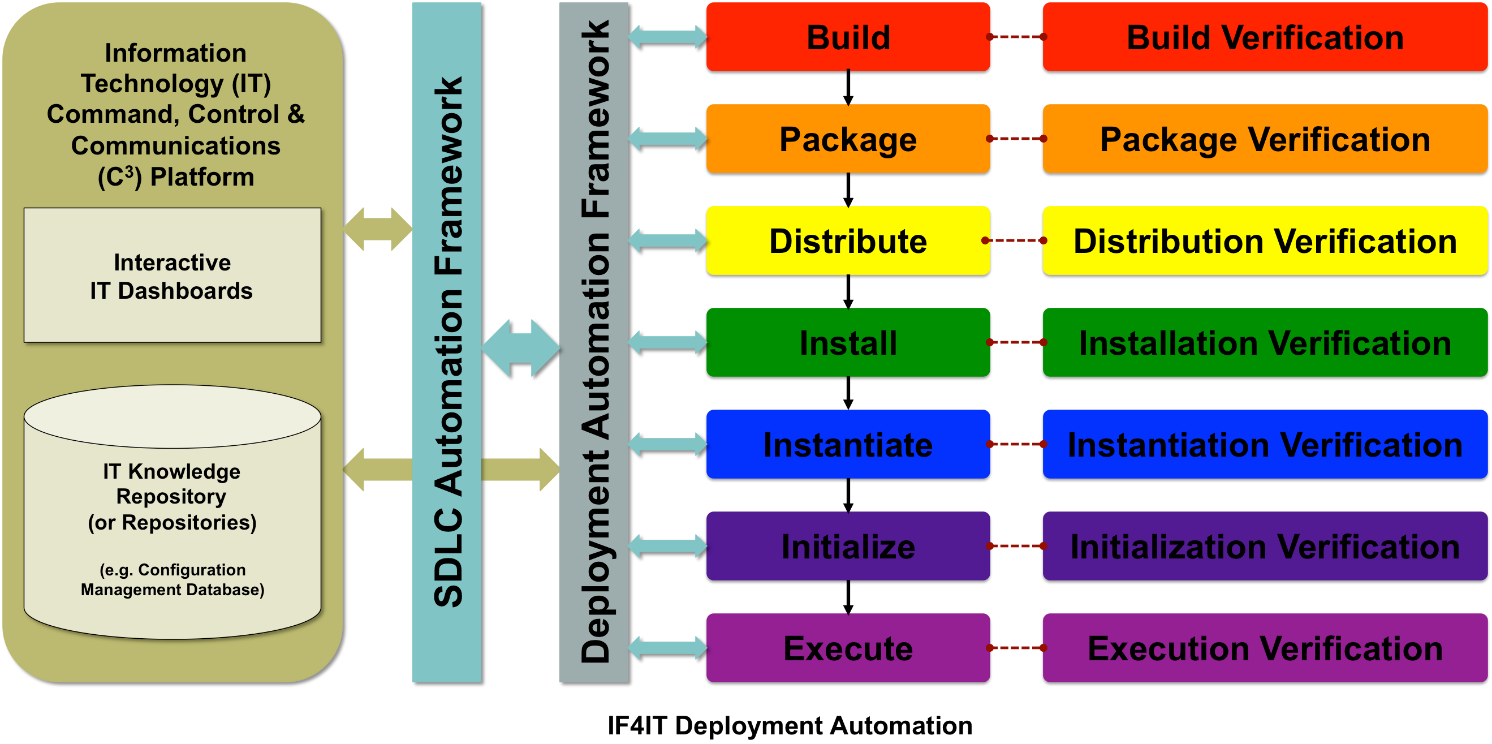
The best practice is simple... Automate! Automate! Automate!

You can always tell a mature IT Organization from an immature IT Organization in that they have built and manage an Automation Framework for which they have strapped in all IT Deployment Functions (i.e. all key activity within and Environment) and all SDLC promotion and demotion functions (i.e. all key activity that allows work to move *"between"* SDLC phases.

The reasons for automation are simple and pretty clear:

* **Clarity:** Automate to decrease delivery time, as trying to coordinate human work is slow, awkward and inefficient.
* **Quality:** Automate to increase quality by eliminating as much human activity, which is prone to error, as possible.
* **Transparency:** Automate to increase transparency by ensuring that automation scripts and tools log and report all details for all activities.
* **Risk:** Automate to reduce Risk, since automation eliminates unforeseen variables that come with human labor as well as since automation ensure rigid repeatability.
* **Repeatability:** Automate to increase Repeatability, since automation requires the development, maintenance and enforcement of strict rules.
* **Costs:** Automate to reduce Costs by performing all of the above.

The Deployment Functions are a part of your assembly line to deliver your solution. No one builds cars, ships, planes, or even computers manually and with no automation. If you and your enterprise are not automating your Deployment Functions, it's time to start evaluating your processes and functions, in order to do so. The automation of your Deployment Functions, through an inter-SDLC Automation Framework, is the mark of a Software Development group that gets big picture Software Development and Delivery.



If your leadership is not pushing hard to automate your IT Organization's Deployment Functions, it's time to start questioning your leadership.

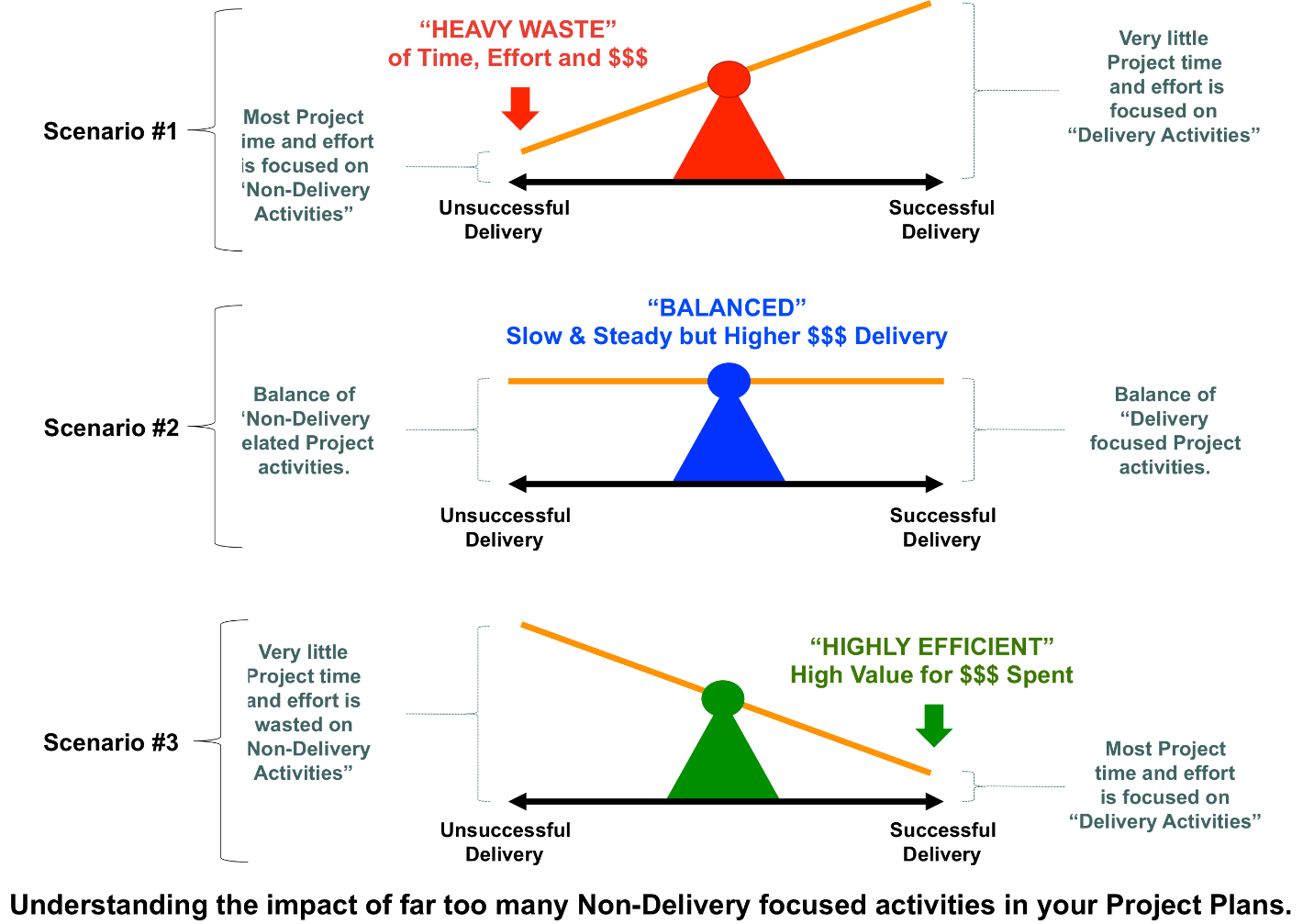
If you, as an IT Professional, are not constantly pushing your leadership and your IT Organization to automate and improve it's Deployment Functions, then it's time to consider or reconsider doing so.

Many Businesses perceive their IT Organizations to be slow, low quality, non-transparent, high risk, and costly. Start to successfully automate your IT Deployment Functions and you'll notice a change in perception.

## How to Leverage the IT Deployment Framework for Better IT Project Plans and More Successful IT Delivery

As pointed out earlier, one of the most common mistakes Organizations make, when it comes to IT Delivery, is to heavily weigh down its IT Delivery Projects with low value tasks or activities that many IT Professionals call "Administrivia." In other words, IT Organizations load their projects with administrative tasks that cater to the desires of Management, Program and Project offices and move further and further away from the types of tasks or activities that are truly related to the delivery of Software and Systems. The facts are simple. The more time IT Delivery teams like software developers and engineers spend on activities that don't focus directly on delivery, the more difficult it will be to successfully deliver IT Solutions quickly, with a high level of quality, and (most importantly) under budget. Smart IT organizations consistently evaluate their IT Program and Project Management processes, always looking for ways to minimize low value activities and maximize their high value activities. The IT Deployment Framework, used correctly, becomes a very powerful tool for ensuring that "High-Value Activities" are identified, accounted for, tracked and communicated. It does this by helping Project Managers, most of whom have very little SDLC based Design, Implementation and Deployment experience, understand what is or isn't important to focus on, when building their IT Design, Development and Delivery Project Plans.

The following diagram highlights three scenarios that are very typical of IT Project Management, across all industries...



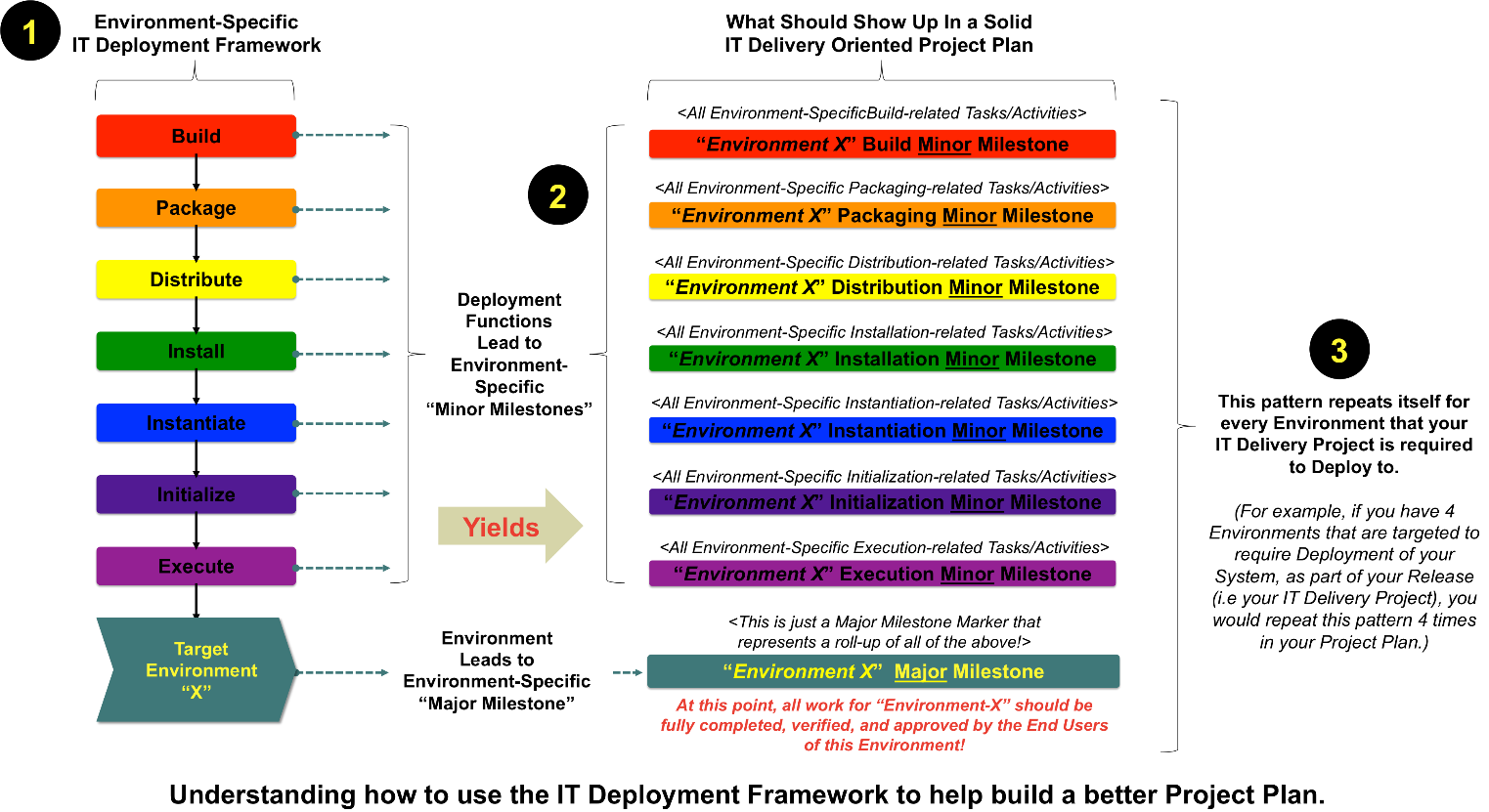
In the first Scenario, #1, IT managers and team leaders load their project plans with far more low-value activities than they do high-value activities. For example, filling out forms to meet requirements for your Program Management Office may seem to be high-value activities for your Program Office but in reality it takes away from valuable time, effort, and money that could otherwise be better spent on higher-value activities, like IT Deployment activities (i.e. Building, Packaging, Distributing, Installing, Instantiating, Initializing, and Executing).

In large enterprises, where there are tens of thousands of people, at a minimum, and where Program Management Offices are pervasive, it's pretty fair to come to the conclusion that the best case hope for efficiency is Scenario #2, which shows a balance of performing work that makes non-delivery stakeholders as happy as possible, while still allowing for Delivery to occur. In this case, Delivery happens but it's far from being as efficient as it could be, such as in a case where there were little to no low-value activities associated with a Project.

The final Scenario, #3, is considered to be the most efficient form of Delivery. In this case, the balance is shifted toward the maximization of high-value actitivities. This final scenario is typical of smaller to mid-sized enterprises, where corporate overhead and bureaucracy is minimal.

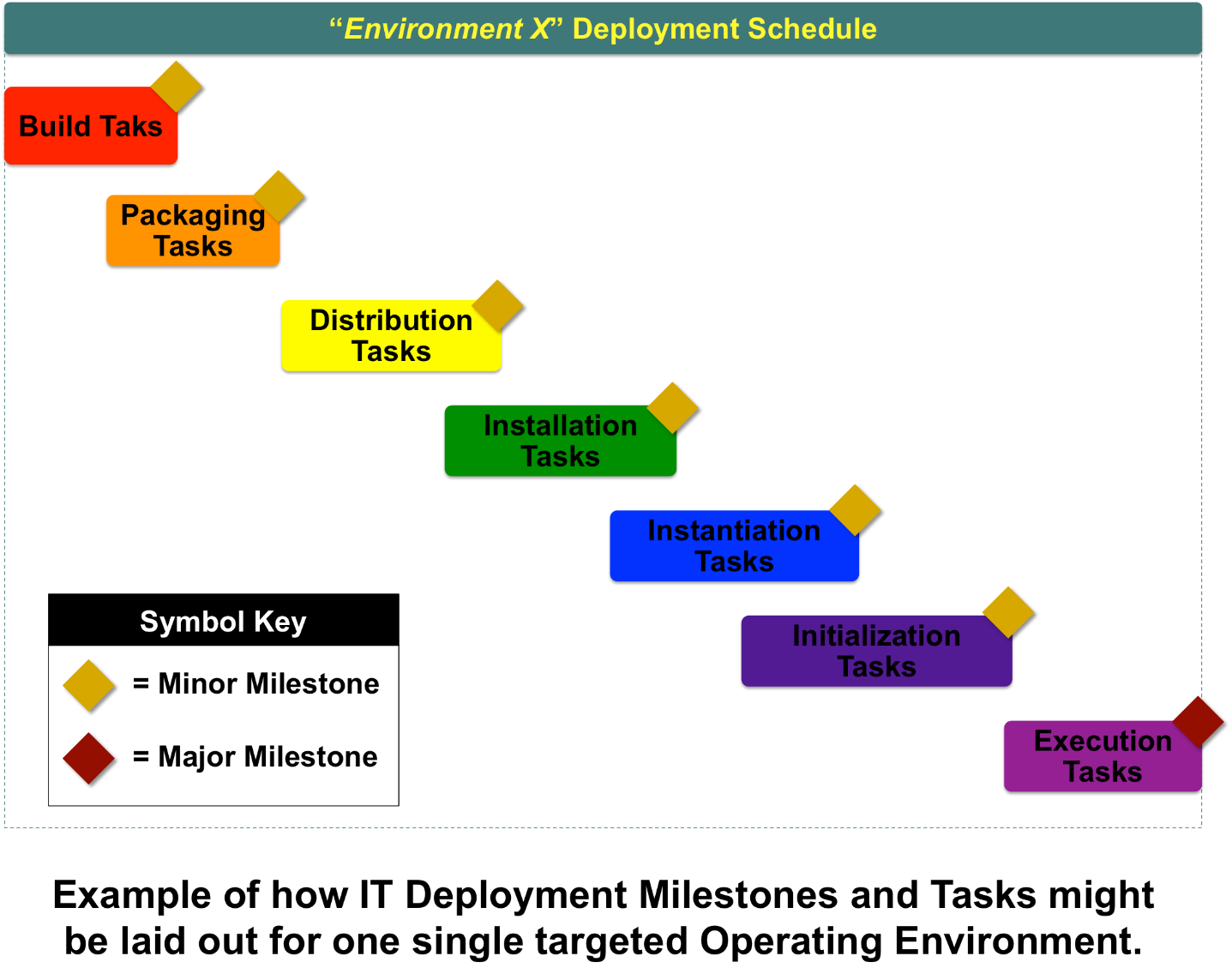
If you're an IT manager or leader in a large enterprise, by looking at the above scenarios you can see and understand why it's so difficult for large enterprises to be nimble, efficient and cost effective. They have organizations like PMOs, Legal, Compliance and Records Management Organizations that are always imposing contraint based activities that are not focused on IT Delivery but more on enterprise administration (i.e. "*Administrivia!*"). The advantage of small enterprises is the lack of bureaucracy and red tape, allowing for focus on activities that are mostly about delivery and not administrivia. In other words, small enterprises have less demand from management and other Organizations that are outside of the immediate Delivery Team to perform what are considered to be lower-value activities and, therefore, are free to deliver fast, frequently, and ultimately at lower costs than IT Delivery teams in larger enterprises who must deal with such impositions.

The following diagram helps understand how smart Project Team Leaders (whether it be an IT Team Lead or the Lead Project Manager on a specific Project) effectively use the IT Deployment Framework to create more detailed and accurate Project Plans...



1. Start with the IT Deployment Framework, which can be broken down into eight (8) key categorical areas for planning (*one for each IT Deployment Function, which equals 7 and one for the targeted Environment, as a whole = 8*)
2. Turn the functional areas of the IT Deployment Framework into "*Minor Milestones*" and the targeted Environment into a "*Major Milestone*." These Milestones represent "roll-ups" or the completion of a categorical set of tasks for that specifically targeted Environment. So, for example, the Build-related Minor Milestone ensures that all Build-related tasks are fully completed for that targeted Environment. The Major Milestone for the targeted Environment represents a "roll-up" or completion of all Minor Milestones for that targeted Environment, signifying that all work for that Environment has been completed, verified, and approved (or Signed-Off) by the End Users of that Environment.
3. Repeat this pattern for every environment that is targeted for deployment as part of the Project. So, for example, you might plan all of these deployment related activities for your Research Environment, your Developer Work Space Environment, your Centralized Build Environment, your Integration Testing Environment, your User Acceptance Testing Environment, your Production Environment, and your Disaster Recovery Environment. In this example, you would repeat the above framework for Project Planning to account for Deployment in seven (7) different environments, spread across seven (7) different sections of your Project Plan. If your enterprise (or even if it's just your System) has a standardized or typical set of Environments that your team will work in, for each and every IT Delivery iteration (i.e. a "Release"), then it makes sense to create reusable Project Plan Templates that lay this framework out so that you and your team won't have to manually fill all of this in, over and over again, for each Release or Project.

The following diagram shows an example of how IT Deployment specific Tasks and Milestones might be laid out, in a Project Plan to support the delivery of a Release...



So, if you think you're done with your Project Plan, think twice! This is just the beginning. For your project plan to be complete, with respect to your IT Deployment Acitivities, you're still going to need to dissect each and every "*Minor Milestone*" into the detailed tasks that need to be performed for each and every technology that your Software or System uses. How do you accomplish this? This is "*exactly*" what ***Detailed Design*** is for.

## Important Summaries and Conclusions

The following represents what the Foundation believes to be a minimum set of summaries and conclusions that an IT Professional should take away, after reading about the IT Deployment Framework...

* It is important to know and understand the definitions of and associated with IT Deployment.
* IT Delivery and IT Deployment are two completely different things. IT Deployment is leveraged, multiple times, by IT Delivery.
* Enterprises who are smart enough to focus on clarifying and improving their IT Deployment activities reap the same types of benefits as industries who constantly work to improve their manufacturing operations.
* The IT Deployment Framework is composed of seven (7) standard and highly repeatable IT Deployment Functions.
* For the Release of Software or a System to be successful, all IT Deployment Framework Functions must be successfully executed and completed in each and every targeted Operating Environment.
* It's important to know, understand and leverage key IT Deployment related Roles and Organizations that can help make an IT Organization more efficient.
* Complete deployment for a specific Technology usually implies the deployment of three categorical sub-areas that include: 1) Business Source Code; 2) Software Infrastructure; and 3) Hardware Infrastructure. A smart IT Delivery Team accounts for all three of these areas, for each and every Technology used in its Software or System Release, across all targeted Operating Environments.
* Standardizing IT Deployment helps yield simple and repeatable operational metrics that lead to powerful transparency into how IT Delivery Teams are performing.
* Each IT Deployment Function is a critical piece of the bigger Configuration Management puzzle. Mature enterprises use their IT Deployment Functions to collect, coordinate, store and analyze most of their Software and System related Configuration Data. Also, each IT Deployment Function is dependent on the IT Deployment Function that precedes it.
* A mature IT Organization not only tests the behavior of its Software and Systems but also ensures complete and successful verification of all IT Deployment Functions, in all targeted Operating Environments.
* Automation of your IT Deployment activities helps decrease the time to deploy, increase the frequency of deployment, increase the quality of deployment, and reduce the costs associated with deployment.
* Organizations that focus on too many "Administrivia" type activities usually struggle to make IT Delivery quick, efficient, high quality and cost effective.
* The IT Deployment Framework can be used to help create very detailed, delivery-oriented Project Plans to help support your Software and System Releases.
* Successful IT Project Plans are very clear about any and all IT Deployment related activities and Milestones, for each and every Operating Environment that IT Deployment is targeted for.

- See more at: http://www.if4it.com/SYNTHESIZED/FRAMEWORKS/DEPLOYMENT/deployment\_framework.html#sthash.gvaj0I53.dpuf

circumventing