

**White Paper**

A Roadmap to Enterprise Platform Services

**Platform Engineering  
Maturity Model**

# Table of Contents

<b>Table of Contents</b> .....	<b>2</b>
<b>Overview</b> .....	<b>3</b>
Modern Platforms .....	3
Platform Standardization & Consolidation .....	3
Shared Platform Organizations .....	4
Platform Extensions .....	4
Certified Binaries & Images Available for Download .....	5
Certified Blueprints .....	6
Certified Recipes, Manifests and Promises .....	7
Certified Platform Services.....	8
<b>The Platform Engineering Maturity Model</b> .....	<b>10</b>
Level 1 – Ad-hoc Platform Sprawl .....	10
Level 2 – Platform Standards.....	10
Level 3 – Certified Platforms .....	10
Level 4 – Certified Blueprints, Recipes & Services.....	11
Level 5 – Continual Optimization.....	11



## Overview

This paper provides an overview of the platform standardization problem that most organizations have, outlines different levels of the Platform Engineering Maturity Model, and helps Platform Engineering groups identify at which level of maturity their organization is today.

## Modern Platforms

Enterprises, ISV's and SaaS providers are significant users of modern software platforms. In general, most of them will first identify with a programming language/model. For example, if you ask a developer what their platform is, they are likely to answer "We're a .Net shop," "We're a JEE shop," (enterprise Java) or "We're best of breed with lots of custom platforms." Alternatively, some will identify with a major product like ERP or CRM and the vendor who provided it: "We're an SAP shop," etc. At most large organizations, the platforms are a combination of operating systems, middleware stacks, databases, open source frameworks and products from all of the common ISV's (MSFT, SAP, IBM, ORCL, etc.)

Although organizations strive to have a single, unified platform vision, this is a constant struggle. In many large companies, multiple parties may have the authority to purchase new platforms, leading to multiple, often redundant, choices. This splintering is a well-known problem to architects and their management. As they strive to provide a uniform solution, they are forced to juggle an increasingly diverse portfolio of potential solutions. This problem has historically been attacked via two means:

1. Platform Standardization and Consolidation
2. Shared Platform Organizations

## Platform Standardization and Consolidation

The most common first step that organizations go through to clean up their platform diversity is to publish a set of standards. Generally, the publications identify which platforms are considered the official enterprise standard. It is common for the standards to identify a platform category with all of the vendors, products, versions, recommended status and trending directions. For example:

### *Category: Relational Databases*

Product	Version	Recommended Usage	Trending Direction
Oracle 9i	R2; 9.2.0.1	Legacy Support Only	Toward 10G
Oracle 10g	R2; 10.1.0.2	All new projects	Toward 11G



Oracle 11g	R2; 11.2.0.1	Experimental; Exception approval required.	
Sybase ASE	15.5	Legacy Support Only	Toward 10G

These publications are often produced by either the enterprise architecture group or by a platform engineering organization. They are made available to all of the engineering teams and are updated on a regular basis (usually quarterly).

In addition to providing standards, many organizations will provide assistance to move development teams to the recommended platform. On occasion, this effort is done through a proactive program known as “platform consolidation.” Here, inventories of legacy or non-supported platforms are identified along with the applications that use them. Assistance is provided to the engineering teams to move their software to the new platform. The assistance typically comes in one of three models:

1. Technical papers, emails, etc. on how the team can do it themselves
2. Consultants/mentors who provide assistance in the migration
3. A migration factory, i.e. a third party does the migration for the application team

## Shared Platform Organizations

Companies attempt to organize themselves to best serve the needs of their internal and external customers. IT groups are typically organized first to serve their internal customers (enterprise functions, divisions, business units, etc.) and second, to serve other parts of IT through centralized shared service centers. With the increase in the number of platforms and their associated complexity, it is now common for enterprises to have an organizational unit whose sole mission is to maintain and mature the common platforms on behalf of their internal customers: the application teams. These groups typically own the platform roadmap and the associated decisions related to sun-setting a legacy platform or adopting a new platform or version.

## Platform Extensions

The shared platform organizations typically add additional functionality to the off-the-shelf platforms. This work is often known as “platform engineering.” Here, canned platforms are customized to better meet the needs of their internal IT groups. These extensions are usually related to integration between platforms. For instance, a middleware platform might be



extended so that it has a direct plug-in to the enterprise identity management system – or perhaps into the corporate performance monitoring system. The goal of these extensions is to make the collection of platforms work as an integrated system, as well as to decrease the amount of work necessary to maintain and operate each individual platform.

## Certified Binaries and Images Available for Download

In addition to specifying and extending the recommended platform vendor, product and version, it is common for an organization to create a “certified” version. This often reduces confusion on which product to use and is typically cached on the local network for quick access. In addition, some mechanism (like a portal) may ask the user to identify who they are and how they intend to use the software. These questions allow the platform teams to better track software licenses and keep the inventory of applications-to-platforms up to date.

The platform organization is responsible for identifying the certified release of a platform (and their extensions) and making it available to the application teams. Historically, this has been accomplished in one of two ways:

1. **Common Download** - Platform teams upload the software to a shared location and provide direction to application teams on how to download and install the software.
2. **Common Image** – A virtual machine image is created with the certified version of the software pre-installed. The image is made available to in the virtualized environment or lab management system (VMware, etc.)

This process allows customers to obtain hardened versions of platforms that may have been extended by the platform engineers. However, this solution runs into problems in a few different circumstances:

1. Complexities related to static images not working together in multi-tiered or complex topologies requiring the use of multiple platforms (web + messaging + database, etc.)
2. Complexities related to topologies changing dynamically based on adaptive IaaS environments. Naturally, this situation is prevalent in all cloud environments.
3. Complexities related to managing multiple configurations across environments (Development vs. Test vs. Staging vs. Prod)
4. Complexities related to tight coupling between operating systems, versions of OS's and the platforms.



Because of these complexities, more mature shared platform organizations are embracing simple but powerful techniques to handle these issues via Certified Blueprints, Recipes and Services.

## Certified Blueprints

The more sophisticated platform builds are typically handled by tooling that specializes in complex topologies, dynamic infrastructure (virtualized & cloud) and deploying to multiple targets (physical, virtual, different OS's, different versions, etc.) Blueprinting software, the next generation of “release management” or “systems automation,” enables the platform organization to deliver platform solutions in a rapid and repeatable manner.

Blueprints are the digital description of a simple platform (e.g., a database on OS) or a complex platform (e.g., web + app server + DB + security, etc.). The blueprint references all of the necessary operating systems, patches, utilities and other requisite software to create a fully workable environment. These descriptions are typically described in a mark-up language (like CML), are versioned and can easily be compared to prior versions to identify deltas. The blueprint also acts as a digital fingerprint for the software, enabling an organization to maintain a historical audit trail of the platform makeup over a timeline.

Once a digital blueprint is created, it acts as the golden description of the design-time makeup of a platform. This blueprint can be read into a system automation/release management tool to execute the actual installation of the blueprint in a layered approach, based on a dependency tree (operating system goes first, then OS patches, then middleware, etc.)

Organizations that use Blueprinting processes and solutions have the ability to execute compliance, standardization and refresh tasks in a cost effective manner. For example, if an organization were to issue a platform element like a Web server, and find out later that it had severe security vulnerabilities, it would be very difficult, without a blueprinting solution, to identify all the systems that were using it, and to automatically update them with the security patches. Organizations must be able to perform “where used” on platform elements, and to also “update where used” to affect the needed changes.

Blueprinting focuses on the digital design time description of a system (Bill of Materials) and maintains a list of all of individual parts (Item Master). It does not address the last mile configuration problem that is commonly addressed by processes and tooling for the automated configuring of platforms either at “boot time” or at “run time.”



## Certified Recipes, Manifests and Promises

From one perspective, modern platforms are extremely agile – they allow you to use them in a variety of scenarios and are highly configurable. Conversely, their highly configurable nature presents a new pain; these configurations must be actively managed. Historically, platform organizations have managed configurations by creating sophisticated scripting solutions or by manually editing configuration files for each deployed platform. Both of these are problematic when an organization begins to scale. The time and money spent on managing configurations becomes overwhelming, forcing organizations to cut back on innovation (new platforms or extensions) because they are so busy managing their current platform configurations. This issue is known as “last mile configuration.”

Currently, several solutions are available that remedy the configuration problem. In each case, a sophisticated configuration system allows a user to perform tasks (such as changing configuration settings) on remote servers. The collection of tasks goes by different names depending on the vendor, but common terms include: “recipes,” “manifests” and “promises.” Regardless of the term, the goals remain the same:

1. Optimize the ratio of administrators-to-managed platforms. Obvious benefits can be obtained if a single administrator can now manage more platforms than previously possible.
2. Increase the accuracy of the administered platform. A primary source of errors is related to incorrect configurations, often due to the copying of configurations from one environment to another (a.k.a. “configuration drift”).
3. Provide runtime configuration data to IaaS. A core concept of IaaS is that virtual devices (e.g., servers) come into existence for a brief period, such as during an auto-scale, and then go away. Modern platforms need to dynamically reconfigure their settings to deal with the temporally changing topologies related to as-a-service offerings.

The combination of blueprints and recipes enables platform organizations to offer much more sophisticated products to their users, at a lower price. Together, they also serve as the fundamental enabler of next-generation platforms designed for running on a cloud model (PaaS on IaaS).



## Certified Platform Services

### Introducing PaaS

Platform-as-a-Service (PaaS), is a style of computing where development components and engines (aka, “platforms”) are continually running in a shared environment and available over the network, on-demand (aka, “as a service”).

PaaS solutions offer the core technical functions needed by application development teams, such as Web platforms, database platforms, messaging platforms and the like. The initial focus for PaaS was the “public cloud” model where all of the platform services were owned and operated by a 3<sup>rd</sup> party (e.g., Amazon AWS, Microsoft Azure, etc.) Externally provided PaaS is a viable solution for many organizations. However, many enterprises have heterogeneous platforms from multiple vendors along with their homegrown extensions, preventing external PaaS from being a de-facto standard. This has led to a growing interest in “Internal PaaS,” whereby a platform engineering organization takes their current platforms and makes them available as managed services to their internal users. In some cases, Internal PaaS is provided side-by-side with External PaaS, where specific usage scenarios are identified for each provider.

PaaS focuses on improving the “quality attributes,” also known as the “non-functional concerns,” of platforms. For example, messaging platforms will continue to send messages (as they always have), but their “as-a-service” design may offer new capabilities such as massive scalability. The new non-functional attributes provide the motivation for adopting PaaS – either from increased agility or from a cost savings/avoidance. For example, in a PaaS database solution, it is common for new non-functional attributes to be added that automatically take care of backup and recovery, usage-based-billing, linear scalability, exceptionally high up-time SLA’s via hot-patching and HA capabilities, etc. Again, from the user’s perspective – the database functions as it always has – but from an administration and operations perspective, the level of effort and associated maintenance costs have dropped dramatically.

Most PaaS platforms are built on top of IaaS (Infrastructure-as-a-Service). IaaS offers the foundational services necessary to auto-scale, auto-balance, auto-recover and more. Platform engineering organizations know that these capabilities are sophisticated and often prefer to not build them from scratch, but rather to use off-the-shelf implementations of these services as the foundation for their platforms. It should be noted that turning “static platforms” into “elastic platform services” requires a sincere engineering commitment. Some organizations are acquiring platform services from their ISV partners (VMware, IBM, Microsoft, etc.) while others



are finding that the vendor-specific solutions fail to run on top of the corporate IaaS infrastructure, introducing unwanted “platform silos.”

Either way, the modern platform engineering organization is tasked with creating, extending and operating a new set of platform services. All of the preceding best practices (the use of blueprints and recipes) are now vital to the cost effective execution of PaaS.

Platform organizations must transition from offering “static platform libraries” to “runtime platform services.” This is a significant transition for many organizations. Fundamentally, it requires that the organization grows more sophisticated skills around runtime monitoring, diagnostics, and root cause analysis. Additional capabilities are required to manage high-uptime services such as rolling releases, hot-patches, SLA management, capacity planning and more detailed platform roadmaps.

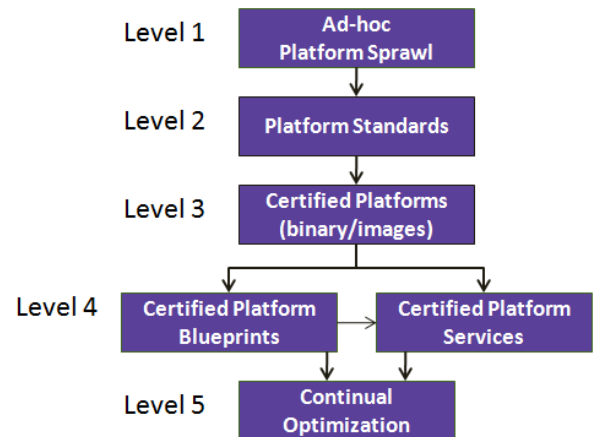
Although these capabilities may sound daunting, they are at the heart of the next-generation software delivery platforms. Fundamentally, they enable application teams to focus on their functional requirements and avoid spending months engineering solutions for scaling, resilience, and so on.



# The Platform Engineering Maturity Model

## Level 1 – Ad-hoc Platform Sprawl

Level 1 suggests an unmanaged and highly unproductive state. Individual application development teams are allowed to pick any platform they desire, regardless of vendor support, in-house knowledge, or other factors. It is common for organizations exhibiting Level 1 characteristics to see high turnover due to frustrated development, testing, and operations teams. Management typically identifies a lack of substitutability of personnel between project teams due to the use of different packages. In addition, it is common for each project team to spend significant amounts of time to ramp up on newly introduced platforms and/or versions. Wasted time and money are the cornerstones of Level 1.



## Level 2 – Platform Standards

Level 2 organizations have acknowledged the chaos found in a Level 1 organization and have taken the first steps to remediation. This typically involves creating an inventory of all of the platforms in use and whittling the list down to a set of recommended platforms. Although this might sound like a simple process, it is highly political and typically takes months to create a platform standards roadmap. A Level 2 organization will work with the application teams to keep the roadmap updated and communicate any changes. It is common for a Level 2 organization to provide some level of “platform governance” where they review proposed application architectures to ensure that they align with the standards. Aggressive companies will initiate platform consolidation programs to more quickly remediate the platform sprawl.

## Level 3 – Certified Platforms

Level 3 organizations have moved beyond the issuing of standards, instead offering the actual platform in a downloadable fashion to their internal customers. Platform engineering teams will pre-certify platforms, versions and builds. Often they will create extensions to allow the platform to more easily work in the local environment. Typically, the downloadable files are available at a central portal for self-service access along with the necessary documentation. The binaries are usually available in either a compressed format (.zip, etc.), an installation file or as an image usable by the corporate virtualization platform.

## Level 4 – Certified Blueprints, Recipes & Services

Level 4 organizations have created processes and employed automated tooling, enabling the creation of platform blueprints and configuration recipes to support the delivery of platform services to their constituents. The organization takes on responsibility for certifying complete topologies and aligns the solutions with the adaptive/elastic infrastructure layer. Strategies are employed to leverage external PaaS solutions (when applicable) and to service enable current platform stacks via the use of blueprints and recipes. Additional capabilities are grown related to platform design, deploy-time configuration and runtime execution and diagnosis.

## Level 5 – Continual Optimization

Level 5 organizations have embraced the concepts of the Level 4, but have also built in closed-loop feedback systems. These systems are core to continual improvement and optimization of the organization. The platform services are continually monitored across a variety of metrics (utilization, cost-per-function, up-time, etc.) and improvement plans are formed and executed. Regular meetings are held with internal customers and stakeholders to identify current issues and forecast future needs. Roadmaps are actively managed and are based on a combination of customer input and lessons learned.

## About

MomentumSI helps organizations mature their platform strategies by using an incremental approach to maturity. At each level, organizations are required to leverage new processes, tools and techniques to increase their efficiency and drive greater customer satisfaction. The Platform Engineering Maturity Model serves as the roadmap for our platform delivery services. For more information, contact [sales@momentumsi.com](mailto:sales@momentumsi.com) or dial 1-888-886-8560.

