CLOUD GOVERNANCE BOARD

FEBRUARY, 2012
SAP CLOUD PLATFORM STRATEGY

BOARD SPONSOR: VISHAL SIKKA

LEADS: NAVIN BUDHIRAJA, JONATHAN HELLER

STATUS: **V1.0, MARCH 30, 2012**

STRICTLY CONFIDENTIAL

TABLE OF CONTENTS

OBJECTIVES FOR THE DOCUMENT

This document summarizes the recommendations from the Cloud Platform work stream on the strategic direction and rationale (what/why) along with the roadmap for SAP Cloud Platform (Platform-as a-Service - PaaS) to support SAP's goal to be #1 player in the cloud in terms of market share and number of users by 2017.

Key Questions Addressed

- What are our goals for SAP Cloud Platform, both for SAP internal applications, and for applications built by customers and partners?
- How will our Cloud Platforms strategy ensure that we do not slow down, but rather support the growth and margin aspirations of SAP cloud business?
- How will we create a differentiated offering of integrated suite of solutions, possibly done on different platforms and technologies– even with M&A (current and future)?
- How will we differentiate against Force.com?
- What is our target cloud architecture including databases, programming models, runtime containers, etc.?
- What are the platforms – SuccessFactors (SF), SAP NetWeaver Neo (JPaaS), ByD Platform, HANA+RDL – for SAP to build new cloud based applications, and guidelines on when to use what?
- How will we build solutions where workflows/scenarios can span across multiple applications?
- What are the architectural and operations requirements to support our cloud strategy e.g. host SAP Netweaver Neo in the same datacenter as SF?
- How much do we invest in each of our current cloud platforms – BYD PLATFORM, SF, SAP Netweaver Neo, HANA?
- Do we migrate our current applications to a different/new platform? If so, when?
- What is the minimum set of requirements that makes SF (or future acquisition) an SAP Solution?
- How do we accelerate cloud and on-premise integration strategy, keeping all options in mind (build/buy/partner)
- What is our mobility platform services strategy, and synergies across SAP and SF?
- What is our collaboration platform services strategy?
- What do we do with our CloudFoundry relationship? Do we create alliances and relationships with other non-SAP PaaS providers like Azure?

Out of Scope

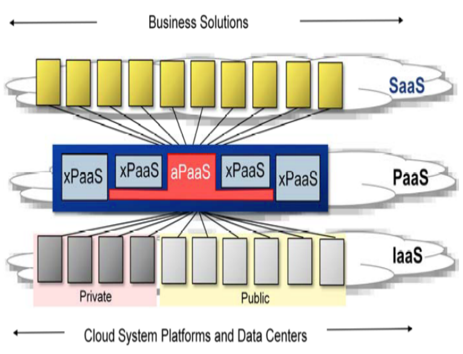
- Direct monetization goals from SAP PaaS and associated business case
- B1 On-Demand platform roadmap
- Analysis of current cost of operations of different platforms
- Ecosystem strategy
- Platform operations
- Detailed Execution roadmap – to be developed subsequently with respective development teams

Target Audience

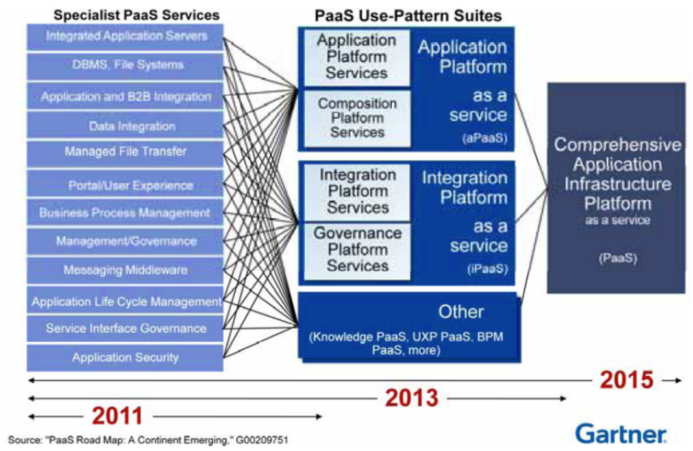
This document is strictly for internal use only.

MARKET OVERVIEW

A commonly agreed market definition of PaaS is “a complete application infrastructure platform” delivered as a service. This includes application platform (aPaaS), integration platform (iPaaS), and other typical on-premise middleware capabilities like BPM, Portals etc. together with development tools, runtime, administration and management tools (Source: Gartner, Forrester).



Source: Gartner


Specialist PaaS Services
Integrated Application Servers
DBMS, File Systems
Application and B2B Integration
Data Integration
Managed File Transfer
Portal/User Experience
Business Process Management
Management/Governance
Messaging Middleware
Application Life Cycle Management
Service Interface Governance
Application Security
PaaS Use-Pattern Suites
Application Platform Services
Composition Platform Services
Application Platform as a service (aPaaS)
Integration Platform Services
Governance Platform Services
Integration Platform as a service (iPaaS)
Other (Knowledge PaaS, UXP PaaS, BPM PaaS, more)
Comprehensive Application Infrastructure Platform as a service (PaaS)
2011
2013
2015
Source: "PaaS Road Map: A Continent Emerging," G00209751
Gartner

Application platform as a service (aPaaS) delivers the functionality of an application platform as a cloud service. It acts as the underlying enabling technology for cloud-based solutions (SaaS). However, beyond aPaaS, a PaaS also delivers many other services (xPaaS e.g. Integration PaaS or iPaaS). Many of these platform services are themselves developed using the same aPaaS.

The PaaS market is currently fragmented. It is expected that by 2015, leading PaaS providers will offer a consolidated, multifunctional portfolio of platform services. The number of the aPaaS offerings in the market will decline over time through acquisitions, mergers and repositioning of some of the providers to different markets. *(Source: Gartner)*

Additional market requirements and use cases are discussed in Appendix Key Market Requirements and Use-Cases

SAP CLOUD PLATFORM GOALS

Business Goals from Cloud Platform

SAP PaaS shall give SAP and its ecosystem (customers, partners, single developers) the ability to develop, extend, run and maintain cloud applications - either standalone or complementary to SAP’s business software offerings. SAP PaaS has the following business goals:

1. Create a platform for SAP development teams to develop cloud applications (SAP runs SAP), along with tools for their customization by professional services, customers and System Integration (SI) partners. These different applications should provide a seamless end-user experience, even if the underlying technologies (e.g. ABAP, Java, Ruby, .NET) are different.

2. Be the platform of choice for partners to build and monetize adjacent cloud application, thus helping SAP offer a more complete solution (and fill in any whitespace). SAP PaaS should differentiate itself from general-purpose PaaS offerings by including application services, integrated life cycle management, end-to-end tracing, OP-OD integration, etc.
3. Be the platform of choice for customers and system integrators to extend their applications (either On Premise or On Demand) created by SAP or partners. SAP PaaS will thus help our customers transition to the cloud, while maintaining their close relationship and dependency on SAP.

The addressable market for #2 and #3, and target customer segments, need to be investigated further and aligned with SAPs overall strategy.

Key Guiding Principles and Boundary Conditions

- Create platform differentiators for business growth and upsell, e.g. leverage HANA.
- Not slow down, but accelerate and support SAP cloud business growth. Not rip and replace, but rather incrementally evolve to the target platform architecture.
- Support heterogeneous programming models so that we can grow both organically and inorganically.
- For the near term, maximize business growth by reducing friction for end-user adoption and upsell, and quickly launching new products. Less focus on optimizing the number of platforms, or migrating existing applications.
- Open+1 Strategy – SAP will not only offer PaaS supporting selected programming models (e.g. JVM based and native HANA) but also create a partner eco-system to allow choice to our customers.

SAP CLOUD PLATFORM STRATEGY - OVERALL

Currently SAP has multiple cloud platforms for internal and external use:

- Business One On Demand (B1 OD) – a SaaS delivery of Business One solution.
- Netweaver Neo (JPaaS) – SAP Java PaaS intended for SAP internal as well as external use by partners and customers.
- SAP Business ByDesign (SME branch) – Business ByDesign SaaS.
- SAP Business ByDesign (LE branch) – A code split of Business ByDesign. Intended as a platform for on-demand Large Enterprise LoB applications. Intended for internal use.
- SuccessFactors Platform – A platform providing a set of shared services used by SuccessFactors applications built on 5-6 different platforms. Intended for internal use.

This list does not include additional SaaS solution running on various Java servers already identified to be integrated with Netweaver Neo, or those inherited through prior acquisitions e.g. Crossgate.

The current SAP cloud platforms and frameworks are designed to be used in a single development environment and primarily expose services (or APIs) to be used locally. There is limited support for heterogeneity of runtime containers, or for remote-enabled shared interfaces and shared business objects that can be available in various technologies.

In order to support sustained growth, we will drive the SAP cloud platform to be more heterogeneous with support for applications written in multiple programming languages (many through acquisitions that we decide to not rewrite because of time and cost reasons). However, to provide a seamless experience to end-users in this heterogeneous environment, hiding the different technologies used in the backend, and to build differentiated applications leveraging the full potential of HANA, SAP cloud platform needs to have the following characteristics:

1. **Shared Data Centers**
   A critical aspect to providing a unified experience is to ensure that the SAP cloud platform is available in all data centers that can host a SAP cloud application. In order to get the best experience (for example, consistent response times across different applications), all applications that a given user accesses should be served out of the same data center.

2. **Cloud Application Platform with Full Use of HANA Capabilities**
   HANA is at the core of SAP Technology strategy and as such it will be a used by SAP SaaS and PaaS. Traditional three-tier architecture treats the database as a "dumb" data store and does not leverage HANA's ability to execute application logic in parallel, without fetching, caching and transmitting large amount of data between the database and the application server. In order to get the full performance boost possible by HANA architecture, the software needs to be designed and built in a different way by pushing the business logic and processing to the data layer.

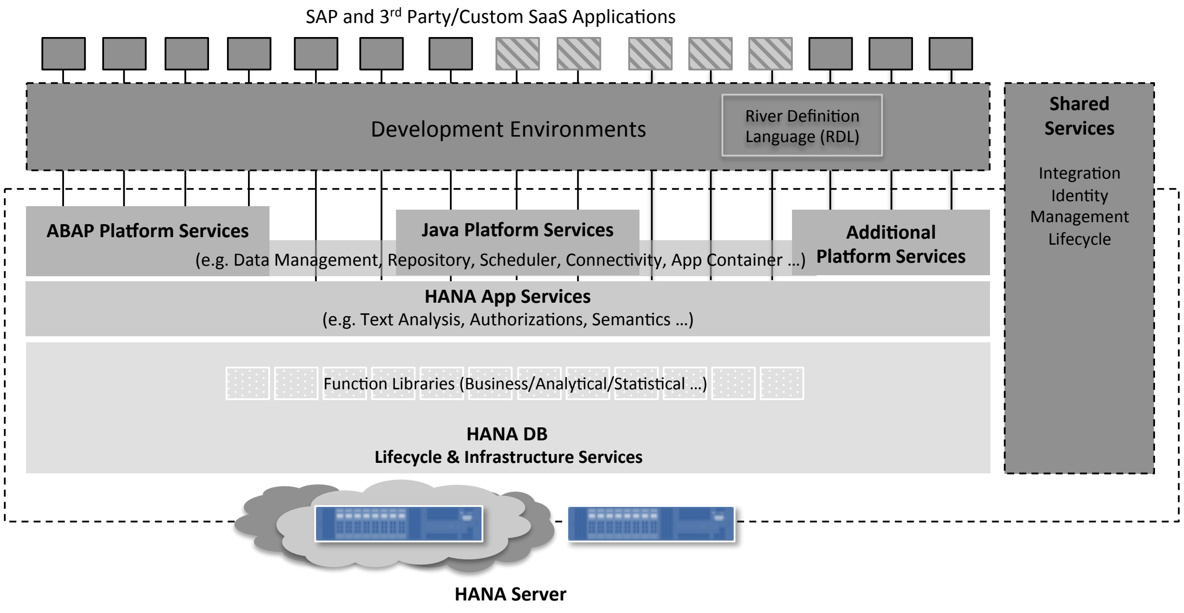
   The cloud application platform should also allow us to build differentiated offerings through optimized cross-applications data sharing. Data sharing between SAP cloud applications and customer extensions will be enabled by direct access to the data layer, honoring access rights restrictions and tenant isolation settings, as defined for each business objects. SQL queries will replace unnecessary replication of data and message based integration methods and will enable new types of data exploration and visualizations. Direct access to the data layer will need to go through public interfaces such as "views" to allow independent lifecycle of each application and avoid a monolithic architecture. This target architecture for Cloud Application Platform and its roadmap is discussed in the section **CLOUD APPLICATION PLATFORM**.

3. **Shared Platform Services**
   Providing a seamless user experience across different applications requires consistency across common functions like Identity, Security, Configuration, Business Objects, Integrations, Reporting and Analytics, Collaboration, Mobility, etc. This will be achieved in the SAP cloud platform by creating a common set of shared services that are available to all applications, independent of the technology that is used to build an application. While enabling a consistent user experience, these shared services will greatly expedite time to market by providing many of the core capabilities needed in an application. The shared services are discussed in more detail in the section **SHARED PLATFORM SERVICES**.

CLOUD APPLICATION PLATFORM

Target Cloud Application Platform – HANA + RDL+ Open

SAP is developing a new programming model and set of domain specific languages - code named "River" or RDL - for application development leveraging the full potential of the HANA architecture as shown in the picture below *(this programming model is not related to River 1.0).*


SAP and 3rd Party/Custom SaaS Applications
Development Environments
River Definition Language (RDL)
Shared Services
Integration
Identity
Management
Lifecycle
ABAP Platform Services
Java Platform Services
Additional Platform Services
(e.g. Data Management, Repository, Scheduler, Connectivity, App Container …)
HANA App Services
(e.g. Text Analysis, Authorizations, Semantics …)
Function Libraries (Business/Analytical/Statistical …)
HANA DB
Lifecycle & Infrastructure Services
HANA Server

River will enable application development per the Timeless Software Principles - separation of the content from the container and separation of the intent from the optimization. This will enable developers to build applications rapidly and in a concise manner and allow the compilers and runtimes to optimize the application to the specific runtimes and configuration in which the application runs.

River will allow developers to express the application business objects and application logic in a **technology neutral** way, along with the flexibility to use other open programming languages (e.g. Java) for technology dependent parts. The technology independent parts will be able to leverage new technologies once River compilers for these technologies are available.

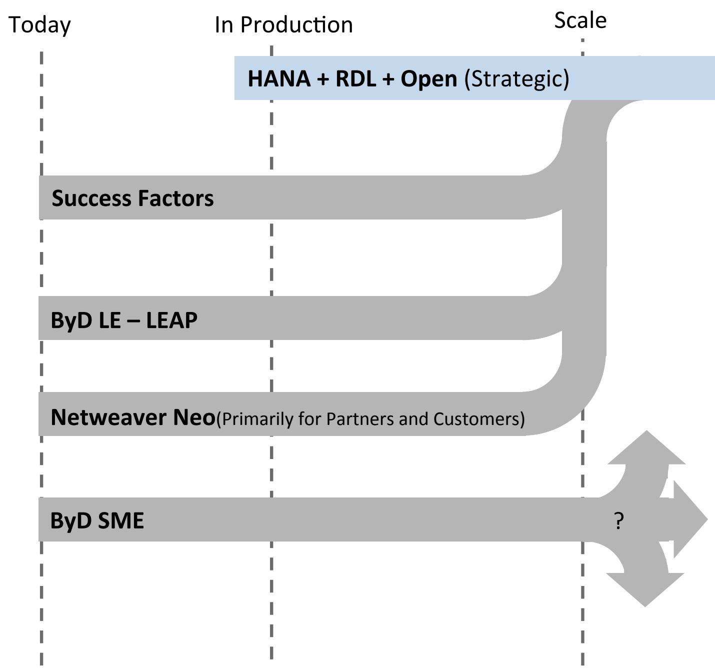
As a general rule, new applications will be built on HANA, leveraging HANA's programming model. However, until River becomes available, a HANA based application should be built using the existing tools and technologies supported by HANA (e.g. HANA Studio and SQLScript), and the available programming models available in the SAP cloud platform (see below). It is then the developer's responsibility to optimize the application to HANA.

Migration of existing applications to HANA+RDL in future will be driven as needed by the application. The goal is to migrate an application to HANA once HANA supports the functional and operational requirements needed for the application.

Existing Cloud Platforms - Strategy and Roadmap

It will take us time before our vision of an integrated HANA+RDL+Open platform (along with shared platform services – see next section) matures. As stated in our guiding principles, for the near term, our focus is on maximizing business growth by reducing friction for end-user adoption, upsell, and quickly launching new products. We will focus less on optimizing the number of platforms, or migrating existing applications. Therefore, in the interim, we will continue with some of our cloud platforms and incrementally evolve to the target platform architecture.

PLATFORM ROADMAP FOR NEW APPLICATIONS


Today
In Production
Scale
HANA + RDL + Open (Strategic)
Success Factors
ByD LE – LEAP
Netweaver Neo(Primarily for Partners and Customers)
ByD SME
?

SAP Netweaver Neo (aka JPaaS) Strategy

- JPaaS will be our application platform for many of our Shared Platform Services (e.g. Integration Service, ID Service) and SAP applications (e.g. Portal OnDemand). However, JPaaS will need to support additional ways of achieving multi-tenancy (e.g. the shared container model used by SuccessFactors applications) to be a platform for SF standard multi-tenant applications.
- JPaaS will be our only offering to SAP ecosystem and customers to host their extensions to SAP On Premise and Cloud applications, including SF applications. It is expected that most customers and partners will prefer an open and standard technology stack. JPaaS provides an easy to use end-user development environment, deployment/testing on the SAP cloud infrastructure, monitoring, management and billing.
- JPaaS also has to become IaaS independent. Currently, JPaaS runs on SAP Cloud infrastructure only and cannot be hosted at any other data center. It needs to be deployable at the same data centers as SuccessFactors applications to be used to extend SuccessFactors applications, while minimizing latency.

SuccessFactors Platform Strategy

- We will not replace JBoss / Tomcat in SF with LJS right now as there is no substantial business value in doing so. We should do so in future (TBD) to drive "SAP runs SAP" message as migration from Tomcat to LJS is expected to be low effort.
- Migration timelines of the underlying databases - Oracle RDBMS and MS SQL - in SF to HANA will be determined based on the results of the ongoing prototype effort.
- We will not move SF to Netweaver Neo at this time for the reasons stated above. Additionally, Netweaver Neo currently does not provide the capabilities needed by SF applications (see details below). Future move should be justified based on business needs.
  - Lack of support for other languages beyond JVM based languages - SF applications were developed in – Ruby, .NET and ColdFusion.
  - Netweaver Neo dependency on SAP Cloud Infrastructure - SF applications need to be able to run in multiple SF data centers (in different countries).
- While it is possible to build new Java-based apps on this platform (as SF continues to, the latest application being Employee Central), we will restrict development of new application on SF platform only to those that tightly integrate with SF OD Talent and HR suite, as that is likely to result in the fastest time to market.
- We will expose some of the SF platform and application services (example session management) via Netweaver Neo so that they can be leveraged by other SAP cloud applications. Once it can make SF platform services accessible, it will be used for new applications that are tightly integrated with SuccessFactors.

BYD PLATFORM Strategy

- ByD Platform (LEAP) should be used as a platform for new applications that are tightly integrated to other ByD Platform based applications or if application is constructed primarily from ByD platform services and existing business content
- Evolve ByD Platform its applications to support the seamless user experience in a heterogeneous environment by
  - ByD Platform to serve as a source of shared services such as incident management and master data hub for entities already available in ByD Platform.
  - ByD Platform to expose its business objects for direct data access by other platforms
  - ByD Platform should be enhanced to support direct data access of data sources maintained by other applications.
- Keep ByD platform for internal use and do not allow partners and customers to extend the platform directly in ABAP or ABAP tools so we optimize the applications for HANA by altering the platform without the burden of backward compatibility. Partners and customers will continue to use BYD Studio (scripting in Visual Studio) until a cross-apps extensibility tool becomes available.

River 1.0 Strategy

- Our current River 1.0 is primarily suitable for the development of small standalone multi-tenant applications. It is heavily restricted and cannot be used to extend or customize existing OD or OP applications. Therefore, we recommend that SAP discontinue River 1.0 and migrate "Carbon Impact" as a native HANA application.
- SAP PaaS attractiveness to customers and SI is also dependent on our ability to offer a rapid application development environment that can be used to build simple extensions to SAP applications, and can be extended further by standard development tools and programming languages.

Guidelines for Choosing a Programming Model and Platform for SAP Cloud Applications

Until the shared platform services and River programming model is available, the decision of runtime environment and programming language is specific to each application considering the following factors:

- Alignment with SAP Technology Strategy and in particular the utilization of HANA architecture
- Time to Market and Development Productivity
- Availability of the platform services and supported qualities (e.g. extensibility) required by the application in the respective platform
- Support for cloud and in particular multi-tenancy environment and the associated cost of operation
- Affinity of the application to other applications developed on the same technology stack based on code reuse, data reuse, ability to use existing local interfaces and similarity of application type (e.g. data entry, analytics). If the application is of a different type, it is an indication to evaluate the option to develop it on a different stack with higher priority.

Prioritization of technology stacks for new internally developed applications (high to low):

1. Native HANA application services (HANA XSEngine based applications)
2. NetWeaver Neo
3. SuccessFactors and ByD Platform

Usage of other technologies has to be approved by the Cloud Technology governance board. Extensions to existing applications such as Jam, Inform or StreamWork are excluded.

During 2012, due to the limited maturity of HANA application services, it is expected that very few applications will be built as native HANA applications. Applications will likely benefit most from SuccessFactors or ByD Platform in terms of time to market.

PaaS Strategy for Partners and Customers

As discussed before, SAP Neo will be our only offering to SAP ecosystem and customers to host their extensions to SAP On Premise and Cloud applications, including SF applications. It is expected that most customers and partners will prefer an open standard technology stack with wide adoption. A critical capability needed to support this would be to enable SAP Neo in other data centers e.g. the SuccessFactors data centers.

Third-party PaaS Strategy

SAP PaaS will differentiate its PaaS offering from other vendors by offering seamless OP-OD connectivity, a rich set of integrated shared platform services, access to HANA technology, end to end lifecycle management. Etc.. However, in line with our Open+1 strategy, we will also support a PaaS partner-ecosystem (e.g. Microsoft Azure, VMWare CloudFoundry, etc.) to provide customers with a choice. We will allow customers and partners to access SAP shared platform services and applications from these non-SAP PaaS. It is recommended that SAP continue its current alliance with CloudFoundry and in addition also consider a strategic alliance with Microsoft Azure.

SHARED PLATFORM SERVICES

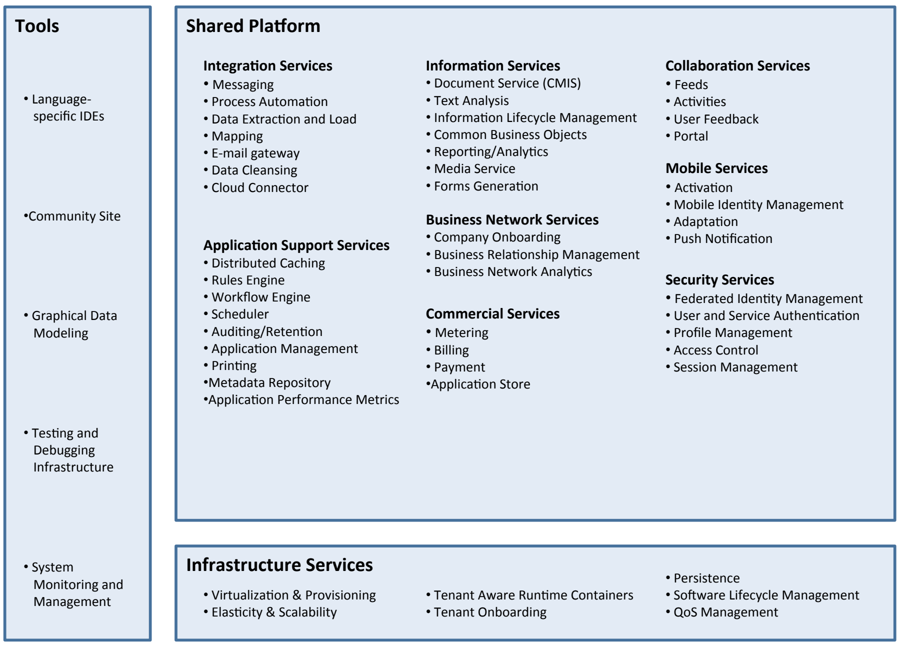
Shared Platform Services Overview

The shared services provided by the SAP cloud platform have the following characteristics:

- Services are constructed on a limited number of technology platforms and development environments.
- Services can be consumed from a (larger) number of technology platforms and development environments, including from environments outside our cloud platform.
- Where possible, services shall be designed with standards-based technology-neutral service interfaces (e.g. Web Services, REST/OData). (Note that remote-enabled interfaces do not necessarily imply remote interfaces. Remote-enabled interfaces can be highly optimized when used within the same VM or address space.)
- For services where technology-neutral interfaces are not appropriate, appropriate APIs will be implemented for the environments from which they are most likely to be consumed.
- There is no distinction between services that are used by applications built on the SAP cloud platform vs. services that are consumed by applications built on external platforms. There may, of course, be access control restrictions.

The availability of the shared platform services as remote enabled with local proxies or optimized implementations in various technologies will make the decision of the runtime environment and the programming model a matter of matching the team skillset, the application operational constraints in term of cost of operations, target frontend technologies, etc.

The following figure shows the Shared Platform Services that will be provided by the cloud platform:



- **Shared Platform** – SAP PaaS supporting application development with a set of services required for enterprise-class applications with consistent product experience. The services are organized into several categories: Integration, Information, Collaboration, Mobile Security, Business Network, Commercial and Application Support.
- **Infrastructure Services** – The components of Infrastructure as a Service, used by the PaaS. This document doesn't discuss on this layer.
- **Tools** – are a set of tools and capabilities available to the application providers, the PaaS operations team and to the customers.
  - IDE – The development environment, mostly eclipse based. The IDE is dependent on the programming language and technology stack used.
  - Community site – A global web site for the PaaS application developers community.
  - Graphical Data Modeling – A modeling tool for business objects extensibility.
  - Testing and Debugging Infrastructure – A sandbox for testing and developing applications prior to deployment and activation for production.
  - System Monitoring and Management – PaaS management console.

In the next three sections, we give brief descriptions of three of these services – *Collaboration Services, Integration Services* and *Mobile Services*. We believe these three are key to SAP's cloud strategy (social applications, seamless OP-OD integration and mobility respectively), and will significantly differentiate SAP PaaS offerings from other vendors.

Collaboration Services

People collaboration capabilities are becoming the basic building block of all modern applications. Collaboration happens in an organization between users of the same application, or users of different applications, or across organizations between users using the same or different applications. We primarily tend to think of the collaboration between users (or individuals); in the future will see more and more automated agents collaborating with humans. The agents may passively participate, recording the collaboration with other users; they may augment the discussion by suggesting relevant information to the discussion at hand, or by triggering collaboration between other individuals (human and otherwise).

The SAP collaboration services should make available the following features to all applications built using the SAP cloud platform.

- **Shared User Profiles** – Allow user profiles to be used across multiple SAP and custom apps
- **Status updates** - Users in same or other applications should be able to provide updates.
- **Groups** - Allow users in applications to create groups to collaborate directly in those applications
- **Feeds** – Users should be able to monitor activity and updates across multiple applications in one place.
- **Content sharing** – Allow users to share documents and media inside the applications. Ultimately OP documents should be shareable as well.
- **Application (Business) Content Sharing** – Enables the embedding on application context in the collaboration medium including one of more of the followings: visualization, live-data, potential participants derived from business context and authorization (access control) information.
- **Application updates** – Applications (agents) should be able to automatically interact with users by sending updates to user feeds and profiles.
- **Rich and configurable security and sharing model** – Allow users to manage permissions
- **Search** – Enable users to search through their feeds
- **Support multiple ways of interaction** - mobile, email, SMS, etc.

- **Public Social APIs** - APIs for feeds, profile, updates, and other components that will allow developers (SAP, partner, and customer) to create new applications with embedded collaboration.

Subsequently, collaboration services could also provide the following (lower priority) capabilities:

- **Out-of-box integration with 3rd party applications e.g. Google docs** – Allow application developers to read and write into a Google Spreadsheet.
- **Integration with other Social networks e.g. Facebook and Twitter** – Allow users to create a unified monitoring of feeds from internal as well as external communities.

Role of Jam and StreamWork

SAP and SuccessFactors have independently created two collaboration applications - SAP StreamWork and SuccessFactors Jam. Both applications expose APIs to enable collaboration within other applications. Jam user experience is human centric while StreamWork is goal oriented aiming to guide the collaboration toward a specified goal. The basic elements of a collaboration scenario, however, are the same for both applications: User Profile, Participants, Feeds, Groups and a Collaboration Instance (an activity, a group or a business object). A collaboration scenario uses collaboration mediums such as wiki and blog and instant messaging. StreamWork adds additional tools such as pros/cons table and SWOT analysis.

Keeping the Jam and StreamWork APIs, and the underlying infrastructure, running while trying to provide a coherent user experience (e.g. one notification, see all the user collaboration activities in one place, manage one's collaborations etc.) poses many challenges and is highly dependent on the user preferred solution (or center of gravity). So in order to enable the use of both solutions, together or separately, the underlying frameworks must be consolidated into one framework. The different types of collaboration activities within Jam and StreamWork can be represented as collaboration templates (e.g. decision making, information-sharing, deal-closing etc.). These templates will initialize the different elements in a certain context and set the user experience for the participants to work in.

StreamWork is an integral part of SAP BI On Demand and is being integrated into SAP Business Suite applications using StreamWork REST API and UI based integration. StreamWork Advanced and Premium addition is used by ~860 users (54 customers, ~16 users per customer).

Jam is already integrated into Success Factors applications, and is the foundation of the collaboration-based eLearning (using Plateau and Jambok). Jam Premium version is used ~630,000 users (53 customers, ~12,000 users per paying customer).

Recommendation

- Choose Jam as the technical framework that StreamWork will use for the shared collaboration services.
- Further evaluation of the ability to refactor StreamWork as a layer on top of Jam containing a set of templates, widgets and server side components.

Integration Services

As discussed earlier, one of the key goals of the proposed cloud platform is to enable applications to be composed out of loosely coupled services on heterogeneous technology platforms, thereby meeting the business goals of rapidly driving growth and maintaining agility. The implication of this is that integration infrastructure needs to play a fundamental role within the cloud platform. Rather than an add-on for integrating pre-constructed applications, integration infrastructure needs to be woven into the fabric of the cloud platform and the cloud application development process in much the same way as, for example, database infrastructure is.

Integration infrastructure will be used within the cloud platform for a variety of purposes including:

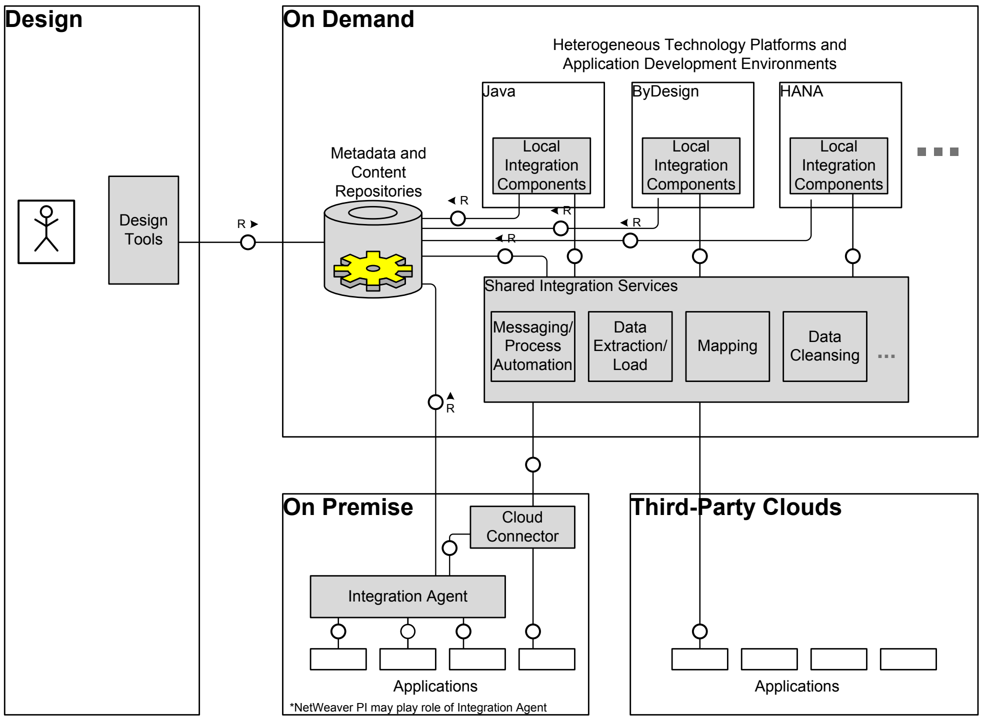
- Creating new on-demand applications from existing shared platform services,
- Integrating SAP on-demand applications with SAP on-premise applications,
- Integrating SAP on-demand applications with third-party on-demand applications or services, and
- Integrating SAP on-demand applications with third-party on-premise applications.

A key challenge will be to meet customer expectations for seamless SAP-to-SAP integration.

Integration covers several different aspects including data integration, process integration and UI integration. In addition, end-to-end monitoring, error handling and lifecycle management also need to be considered as part of the integration infrastructure. In this document, however, we only consider the data and process integration aspects.

Integration Infrastructure Overview

The following figure shows the target state of the integration infrastructure within the cloud platform. As stated above, the focus is on data and process integration infrastructure. The various services and engines in the integration infrastructure should share a common (ideally Web-based) design environment, from where a user can configure, deploy to and manage a distributed set of services, including some that may be on-premise. The infrastructure supports integration among services and applications within our own cloud, integration with on-premise applications as well as integration with third-party clouds.


Design
Design Tools
On Demand
Heterogeneous Technology Platforms and Application Development Environments
Java
ByDesign
HANA
Local Integration Components
Local Integration Components
Local Integration Components
Metadata and Content Repositories
R
Shared Integration Services
Messaging/ Process Automation
Data Extraction/ Load
Mapping
Data Cleansing
...
On Premise
Cloud Connector
Integration Agent
Applications
*NetWeaver PI may play role of Integration Agent
Third-Party Clouds
Applications

Each of the components shown above is discussed in detail in Appendix Integration Services - Details.

Current State and Recommendation

Currently SAP does not offer an on-demand integration product, and SuccessFactors' integrations with on-premise applications are primarily carried out through file transfer, although a partnership has recently been initiated with Boomi.

- The Avatar project within SAP is developing an on-demand process integration product that is due for limited deployment later this year as a JPaaS service. Apart from the integration engine itself, it also provides a lightweight highly scalable multi-tenant runtime that is a good candidate to form the basis for the shared integration services described above. The design tools for Avatar will be based on the tools currently available for the on-premise product, which are implemented in the Eclipse development environment.
- The Business Objects Data Services team has recently initiated a project to create an on-demand version of their product, which is primarily targeted to handle the use cases for HANA-based on-demand applications. Architectural decisions and product plans are still being made.
- The ByDesign platform includes a framework called Process Agent Framework and a capability called Lean PI, which roughly correspond to the ByDesign Local Integration Components described above. Their design tools are integrated with the ByDesign design tools.
- HANA/RDL has the capability to automatically generate REST interfaces from RDL objects, which once again corresponds to the Local Integration Components for this environment.
- MDRS is the ByDesign metadata repository and is also used within the NetWeaver environment. More importantly, it contains the definitions for business objects and process models on several layers that can be the basis for SAP-provided content as described above.
- Nestor is a project focused on semantic collaborative mapping and is now past its prototyping stage.
- Crossgate is an on-demand message hub operated by SAP. It is focused on B2B scenarios.

Given the initiatives above, the proposed roadmap is to focus the immediate efforts of developing the shared integration services around the Avatar project. The Data Services On-Demand project should align itself with this effort; in particular making the various data services inter-operate within a process integration scenario and creating an integrated design time user experience.

Mobile Services

Similar to a cloud application platform for SaaS applications, SAP cloud platform needs to also include support for mobile applications – both applications delivered by SAP, and mobile applications that are built by customers and partners. Described below are some of the key services that need to be enabled by such a mobile platform (or *MPaaS*) – detailed description and roadmap is provided in a separate document.

- Device Management – This includes support for secure activation of a device, and device de-activation in case of loss or theft. Additional capabilities like device locking, application distribution, etc. should be supported as well.
- Mobile Identity Management – Ensure that the identity of the end-user on the mobile device can be securely established. Typically, this will require support for mobile SSO.
- Adaptation – The mobile platform will need to provide access to data and services from other applications (SAP or non-SAP), both OP and OD using RESTful APIs. However, the needs of a mobile application would typically be different from those of browser-based SaaS applications (e.g. to minimize bandwidth and latency). For example, a mobile API that provides access to user's profile might need to collect information from the core-HR system, as well as a collaboration application to merge the user's social profile data. It is more efficient to do this

adaptation inside a mobile service versus expecting the mobile device to make multiple calls to collect all the details.
- Push Notifications – Support for device specific push notifications, where the push can be initiated by SAP or non-SAP applications, both OP and OD.

Shared Platform Services – Roadmap

Shown below is the recommend roadmap for the shared platform services. The services mentioned in brackets (e.g. Document Service under Technical Must Haves in Information Services) are for an external PaaS (for customers and partners) and not required in a PaaS for SAP internal applications.

Category	Must Have for Application Development	Must Have for Production	Beyond
Infrastructure Services	• Runtime Containers • Persistence	• Tenant Onboarding • Software Lifecycle Management	• Elasticity and Scalability • QoS Management
Security Services	• Access Control	• Federated Identify Management • Authentication • Profile Management	• Session Management • (Auditing Retention)
Information Services	• Common Business Objects • Reporting/Analytics • Text Analysis • (Document Service)		• Document Service • Media Service • Information Lifecycle Management • PDF/Adobe Forms
Integration Services	• Messaging • Email	• ETL • Mapping • Process Automation • Cloud Connector	• Data Cleansing
Commercial Services		• Metering • Billing	• Payment Application Store
Collaboration Services	• Portal API Specification • Feed & Activities	• (Mobile API)	• Portal API
App Support Services	• Scheduler • Printing	• Application Management • Business Configuration Services	• Distributed Cache • Rules Engine • Workflow Engine • Auditing/Retention
Mobile Services	• Device Management • Identify Management • Adaptation		

Category	Must Have for Application Development	Must Have for Production	Beyond
	• Push Notification		
Development Tools (Language Specific)	• IDE • Test and Debugging Infrastructure • Community/Wiki		• Graphical Data Modeling • Central Meta-data Repository
Operations Management Tools		• Systems Management and Monitoring • (Single Activity Trace) • (Solution Manager Integration)	• Application Performance Metrics

NEXT STEPS

- Establish a Cloud Technology Governance body (Resolve potential overlap with existing Platforms governance team)
- Develop execution roadmap for Shared Platform Services – further definition and how we would develop them. Options include:
    - Refactor existing ByD capabilities into shared services and enable reuse e.g. service desk
    - New services on JPaaS, HANA + RDL,
    - Leverage existing SF services on JPaaS
- Create target cloud platform investment plan and execution roadmap
- Start JPaaS pilots in the following areas (to be prioritized) to determine requirements and roadmap
    1. SF customers building custom extensions to SF applications on JPaaS
    2. Building a sample SF application on JPaaS (assessment only)
    3. Building a SF platform shared service on JPaaS
- Determine JPaaS roadmap for the following:
    1. A functionally rich PaaS for third-party applications - A platform for partners and customers, including features which will make it attractive to SAP customers (e.g. LCM integration, SAP Passport). Also include ecosystem related services (app-store which supports technology components such as integration components, billing infrastructure for usage based access etc.) and make JPaaS IaaS independent.
    2. A PaaS for SAP and SF apps - supporting more multi-tenancy models and supporting RDL as an alternative to JPA (see Appendix Data Level Multi-tenancy Models on data level multi-tenancy).
    3. Making JPaaS into our Strategic PaaS - Java independent to support specifically HANA XSEngine and arbitrary language runtime (e.g. Ruby).
    4. Making JPaas Open to Partners PaaS - Expose platform services (e.g. Integration as a Service, SAP ID, Document Store) for external consumption by applications running on partner's platforms with the commercial support in-place (security, billing etc.).

APPENDIX

Key Market Requirements and Use-Cases

PaaS by SAP should give SAP and its ecosystem (customers, partners, single developers) the ability to develop, extend, run and maintain applications, either standalone or complementary to SAPs business software offerings. The addressable market needs to be investigated and aligned with SAPs overall OnDemand and On-Premise strategy including who the stakeholders are (partners, customers, single developers, SAP), and which of their demands we want to fulfill to further expand SAPs success.
Based on such a stakeholder analysis, we can further refine what the platform(s) should be, which additional services they need to provide, additional programming language support, etc.

Key Differentiators:

The SAP PaaS will combine Infrastructure (InMemory DB, File system, hardware, servers, network) and shared platform services like identity management and user distribution, integration, mail, forms rendering, integrated analytics, collaboration features, social integration, incident management, etc.. It will also provide rich and extensible application services (business configuration, master data, management of organizational structures) and complete applications (e.g. Sales Order Processing, Financials Accounting). The latter serves as a library of business functionality that enables SAP, partners and customers to extend and orchestrate available business functionalities to new processes and scenarios. This allows us to offer value added differentiation on application-enablement level beyond pure technical components and platform services.

Beyond following other PaaS providers, we need to identify and provide key differentiators, which will help stakeholders to decide to use SAP PaaS instead of other existing or materializing PaaS offerings; for example, publication and distribution of Apps through SAP store including revenue sharing model, tight integration with SAP identity management, high-responsive support, and ease of integration with other SAP products.

Use-Cases for Cloud Platform

The SAP cloud platform is expected to support use cases where OD applications can share process and data with other OD or OP applications, and these applications can be provided by SAP or by third parties. It is possible that these different applications are built on different technology platforms, but they still need to be able to provide a seamless user experience to the end user. Business objects should be sharable between different applications to provide a tightly coupled experience (when needed), This sharing can be “in context” that allows data from one application to be rendered completely within another application – even allowing the host application to redefine the user experience as applicable.

Outlined below are example integrations that the cloud platform needs to support

- Data Integration between OP and OD (OP → OD)
  - OP SAP HCM and OD SF Performance and Compensation. Employee profile information is maintained in the SAP OP application, but used (read-only) within the OD performance and compensation applications.
- Data Integration between OP and OD (OD → OP)
  - Compensation and performance rating data from SF OD sent to OP SAP HCM. Once the performance and compensation cycle is completed, that data is sent back to the OP HCM systems (e.g. to the payroll systems)

- Process integration between OP and OD (OP → OD)
  - Integration between OP CRM and OD Sales. Customer information maintained in the OP CRM systems is looked up on demand by the sales application.
- Process integration between OP and OD (OD → OP)
  - Candidate to hire integration between SF Recruiting and SAP HCM. Once the candidate is hired, data about the new employee is passed to the OP HCM system for onboarding, etc.
- Data/Process Integration between SAP OD → OD
  - SF Employee Profile and SF Jam. Sales OnDemand and Employee Profile. In this case, employee profiles is maintained in an OD system (e.g. SF employee central), but used in context by Jam and a Sales OnDemand application.
- Data/Process Integration between SAP OD and third-party OD
  - Employee profile and LinkedIn. Employee profile is enriched using data maintained at LinkedIn.
  - Recruiting and Background checks. The OD recruiting application needs to perform a background check by a third-party before the candidate can be hired.

In addition to the integration scenarios described above, a cloud platform should also support the following:

- Extensions and Mashups by customers and partners
  - Extend/customize the business objects and functionality e.g. add customer specific fields to existing objects, add new objects, extend functionality etc...
  - Display Employee profile data in a custom maps application developed by customer/partner
- Integrated Reporting and Analytics
  - Reporting across two or more OD applications. Example, report across OD performance and OD variable pay.
  - Reporting across OP and OD applications. Example, report across an OD sales application and an OP CRM application
- Integration with other SaaS Players (e.g. recent SF acquisition with Taleo, Sales OD with Savo)

Data Level Multi-tenancy Models

The multi-tenancy model an application uses is influenced by three factors:

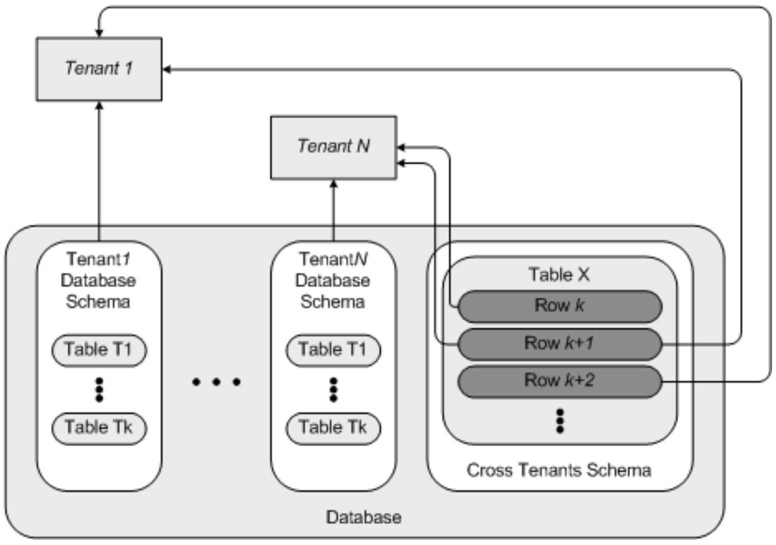
- Compatibility with the SaaS operation requirements such as the ability to restore a single tenant to a previous state independent of the other tenants.
- Information sharing between tenants common in collaboration scenarios where users from different tenants share data with each other.
- Compliance with legal regulations. In certain countries, certain types of data such as personal information have to be stored according to the country's legal regulations. A strict isolation requirement can span from a separate database schema and application processes through separation of physical files to a separate virtual or physical infrastructure.

SAP and Success Factors applications are designed in a way that a multi-tenancy approach on a data level is applied to the application as a whole. All of the application's data follows the same rules. JPaaS enforces the same multi-tenancy model (schema based tenancy) to all of its applications. An application can refine the model and create additional level of separation within its own domain (e.g. separation of the data of different groups of users to different database tables).

The assumption that a single tenancy model applies to the whole application doesn't apply for applications which need to isolate some of the data (for operational or legal reasons) while for other types of information the functional requirements require data sharing and there are no legal constraints.

An approach which splits the application into two or more mini-applications, each with its own multi-tenancy model complicates the operational aspects such as a consistent data restores, and introduces additional challenges in the case of queries that join data from two or more types of multi-tenancy.

A cloud platform should allow the specification of multi-tenancy approach per business object at design time and should support the overriding of the information as part of the deployment process. Applications which access data of the same tenant managed by other applications should be able to retrieve the data following the same multi-tenancy rules used by the data owner.



In the diagram above, tenant 1 and tenant N have their own schema for tenant private data and share a common table for data accessible to both tenants.

HANA + RDL - Details

River is a set of domain specific languages designed for:

- Capturing the semantics of applications: Entity structures, relations between entities and actions that entities can perform.
- Describing implementation of these actions.
- Describing projections (views) of entities.
- To be independent of programming language specific programming model, concepts, libraries and runtime.

JPaaS - Details

SAP Cloud Strategy relies on the existence of a cloud platform for customers and partners to extend SAP On Demand and On Premise applications. SAP ecosystem is looking to understand its role in SAP Cloud and without a place for our partners to extend SAP applications we will push them to our competitors' platforms and will weaken our SaaS applications. SAP Cloud PaaS will also help our customers' transition to the cloud while maintaining their close relationship, dependency on SAP systems and lessen the up-sale opportunities of our competitors to SAP customer who chose their platform.

JPaaS primary objective is to be a PaaS for SAP ecosystem and customers to host their extensions to SAP On Premise and On Demand applications. In addition, JPaaS objective is to host SAP standard multi-tenant applications. Several SAP applications and on-demand services are currently being developed on JPaaS, for example Integration as a Service and On-Demand Portal.

JPaaS, however, has to become IaaS independent so that it can be installed at the same data centers as SuccessFactors (see latency considerations mentioned above). Only then it can be used to extend SuccessFactors applications.

JPaaS, however, will not be the exclusive platform on which SAP applications can be extended. SAP will partner with others to offer extensibility capabilities for non-SAP customers and for those who prefer to develop in programming languages not supported by JPaaS.

SuccessFactors Platform - Details

Below is a subset of the capabilities provided in the SF platform, and we believe these capabilities should be made available in any cloud platform used to build SAP OD applications.

Shared Container Multi-tenancy – SF platform supports the shared container approach to multi-tenancy that allows different tenants and different applications to share the same Java container, while still maintaining the segregation of data in the database. For highly scalable applications, shared tenancy provides significantly better utilization of data center capacity.

Technology Neutral and Service enabled – SF platform services are exposed as technology neutral services that allow them to be consumed uniformly in both Java and non-Java OD applications (non-Java support in an SAP platform is critical to support non-organic growth in the SAP OD offerings). This allows different applications to provide the same seamless user experience, as well as share data between different applications (see below).

Extensible and Customizable Common Objects – SF platform is designed to allow sharing of common business objects between independent, but collaborative, OD applications using technology neutral APIs. For example, profile data can be shared (read and write) between Employee Central and Jam. In addition, these shared objects are customizable and extensible for each customer, a critical capability in a multi-tenant platform.

Integrated Reporting and Analytics – SF platform provides integrated reporting and analytics (using the Inform tool) across all the OD applications that are built using the SF platform, and even applications that are built on other platforms. This not only provides a consolidated view on all data, but also provides access in a manner that preserves the unique security needs of each application.

ByDesign Platform (LEAP) - Details

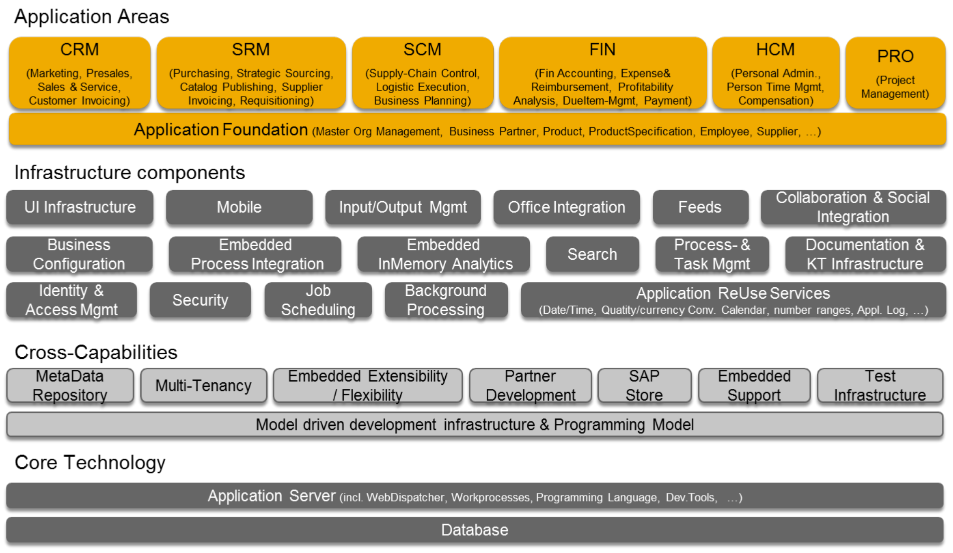
NGAP + ByDesign (ByD) started the foundation for SAP next generation Suite eight years ago. It was designed to be ABAP based, fully Service Oriented, modular and with rapid development environment support.
SAP repurposed the solution to be an On Demand Business Suite making it multi-tenant and split the solution into three layers: Technical platform (i.e. Next Generation Application Platform – NGAP), ByD Platform and the application layer.

ByDesign platform is a PaaS it is operated as a service, which a range of services such as onboarding, provisioning of systems and tenants, software changes, upgrades, monitoring, incident management and problem resolution and more.

ByD Suite (Platform & Apps) suffered from poor performance & high TCO. The solution went through several iterations to reduce the TCO and improve the performance.

Recently, the code was branched and to allow the evolution of two product lines, one for the SME Business Suite market and one for large enterprise line of business applications (aka LEAP). The technical platform layer (NGAP) and ByD Core Foundation remains the same for both branches.

ABAP Server scalability is bound by main memory and its assumptions about database access patterns and connectivity. The ABAP server is OS process-based server. The optimal number of work-processes is based on the number of CPUs and the memory allocation of each process is the same and is determined by expected the peak memory consumption. Work processes use shared memory to store the session state. Each work process maintains its own database connection, used to process all requests regardless of the tenant.


Application Areas
CRM (Marketing, Presales, Sales & Service, Customer Invoicing)
SRM (Purchasing, Strategic Sourcing, Catalog Publishing, Supplier Invoicing, Requisitioning)
SCM (Supply-Chain Control, Logistic Execution, Business Planning)
FIN (Fin Accounting, Expense& Reimbursement, Profitability Analysis, DueItem-Mgmt, Payment)
HCM (Personal Admin., Person Time Mgmt, Compensation)
PRO (Project Management)
Application Foundation (Master Org Management, Business Partner, Product, ProductSpecification, Employee, Supplier, …)
Infrastructure components
UI Infrastructure
Mobile
Input/Output Mgmt
Office Integration
Feeds
Collaboration & Social Integration
Business Configuration
Embedded Process Integration
Embedded InMemory Analytics
Search
Process- & Task Mgmt
Documentation & KT Infrastructure
Identity & Access Mgmt
Security
Job Scheduling
Background Processing
Application ReUse Services (Date/Time, Quatity/currency Conv. Calendar, number ranges, Appl. Log, …)
Cross-Capabilities
MetaData Repository
Multi-Tenancy
Embedded Extensibility / Flexibility
Partner Development
SAP Store
Embedded Support
Test Infrastructure
Model driven development infrastructure & Programming Model
Core Technology
Application Server (incl. WebDispatcher, Workprocesses, Programming Language, Dev.Tools, …)
Database

Existing platform (business) objects and reuse services are exposed SOAP & REST (OData). The public interface is maintained as a Public Solutions Model and can also be deprecated if required. Entities can be released for extensions in partner Add-ons, which can be published and sold through the SAP Store. These Add-ons are developed in a proprietary scripting language and are compiled to ABAP.
Applications built on top of ByDesign can use the same extensibility mechanisms. Customers have specific extensibility options as well such as adding customer specific fields end-to-end from UI to persistence; customize existing forms and reports and adding their own forms & reports for printing and email.

The current UI framework is based on Silverlight. Support for HTML5 based on the NGAP SAP UI5 library for specific scenarios is in scope of the current feature pack. Mobile UIs are displayed using device specific players . The option of developing a native mobile app is also available.

From a developer perspective, ByD Platform offers services such as built-in multi-tenancy, user management, authentication and authorization mechanisms out of the box. Business data is modeled in business objects, which are executed by the enterprise service framework as the runtime container (container managed persistence). The runtime framework handles caching, transaction and persistency. Search and Analytical Views can be added and combined into analytics reports based on operational data (integrated analytics). In addition, the platform injects offers cross-cutting concerns into the application logic such as extensibility and supportability.

ByD Platform architecture assumes that all the application logic is developed in ByD Platform technology stack (which consists of the ABAP applications server, a database server and a UI framework on the client). It is not optimized for a heterogeneous environment in mind since it relies on remote service calls as the integration method. ByD Platform treats other technologies as an Application-to-Application integration scenario based on web services (SOAP/REST).

For example, it does not support sharing of business object models (called Public Solutions Models) for data level integration and its current UI framework is not designed for UI mesh-ups where the UI container is not handled by ByD Platform.

Integration Services - Details

Shared Integration Services. The shared integration services form the core of the integration infrastructure. These comprise of messaging and process automation engines as well as data extraction, transformation and load capabilities. Additional services like data cleansing are also included here. These services are accessed via message- or service-based interfaces and are hence accessible from any technology platform within or outside of the cloud platform. These services are built on a lightweight dynamically scalable tenant-aware architecture of which SAP's Avatar is an example.

Local Integration Components. Local integration components are needed in each technology or application development environment to enable the creation of message- or service-based interfaces in a manner that is appropriate to the environment in question. The level of sophistication of these components may vary from environment to environment from simple API frameworks or stub generators to actually carrying out integration functions like mapping or sequencing. Reasons for having the latter within local integration components include the need to have access to local transaction context, the need to access data that is not easily replicated or the need to create interfaces that do not expose internal details of the applications. However, when this is done, it is important to ensure that the design time user experience is integrated with the design of the overall integration scenario.

Cloud Connector. The primary function of the cloud connector is to ensure secure communication between the on-demand and on-premise environments. In addition, it can include support for higher level concepts like single sign-on. The cloud connector should offer a choice of secure communication options like reverse proxy, tunneling or polling to suit the security policies of the customer. Ideally, it should be installable on premise as downloadable software or virtual appliance, pre-configured with the appropriate on-demand settings (tenant id, etc.)

On-Premise Integration Agent. The on-premise integration agent is an optional component that is responsible for carrying out integration tasks that are best carried out on premise for performance or security reasons. For example, it may be desirable for a large database extract to be performed on premise before the results are sent to the cloud. Although this agent runs on premise, it should be managed and configured from the cloud in the same manner as the on-demand components.

Metadata and Content Repositories. These repositories serve as the bridge between the design and run times. Apart from repositories to store content generated by a user during the course of designing an integration, SAP has the opportunity to distinguish itself is by offering repositories of its own business content that captures best practices and simplifies the integration process. Over the years SAP has gathered insights into the structure and data within common business objects and processes, which can be used as the basis for integration. SAP-provided content can include metadata and actual data for business objects as well as templates for common cross-application business processes. Aside from easing the integration problem, this type of content can attract and bind users to our platform.

Another source of content is the aggregated insights of the user community as a whole—something that is possible with a cloud-based integration infrastructure that is not available in traditional on-premise integration products. At its simplest, this can take the form of shared and searchable repositories of content that users voluntarily make available to others. More complex possibilities include tools like semantic analyzers that derive semantics by mining a large number of scenarios (subject to user permission, of course) and offer suggestions while a user is designing mapping rules or integration flows.

UI Integration

UI integration refers to the ability to create user interfaces that combine information from several applications without necessarily integrating the backends via process or data integration. The cloud platform will support UI integration in two ways:

- Via a Web-based UI framework that allows "mash-ups" of Web-based applications while preserving context across the components
- Via portal services that allow applications to be composed within a portal backend before being presented to a user.

Error Handling

Error handling is a critical issue during integration because integration typically happens in the background, without direct interaction with a user. Moreover, because different systems are involved in integration, it may not be possible to automatically undo actions taken in some systems before an error occurs. In this case, the input of a business expert may be required to resolve an error.

There is no simple service that can handle errors, but a combination of tools, design guidelines and implementation practices can help avoid or resolve errors quickly and keep the cost of operation manageable.

Examples of tools include logs, monitoring consoles and task handlers that route errors to appropriate users. Examples of design guidelines and implementation practices include the Forward Error Handling concepts designed for NetWeaver and ByDesign.

www.sap.com

The Best-Run Businesses Run SAP™