Building SaaS Products

The Cloud, PaaS, IaaS and Service Delivery Platforms (SDP)

Kim Horn

Product Manager

What is SaaS ?

"Software deployed as a hosted service and accessed over the Internet" – Microsoft

Vendor hosts all of the program logic and data and provides end users with access to this over the public internet via Web screens or Integrated via Web Services API's. Usage is On-Demand.

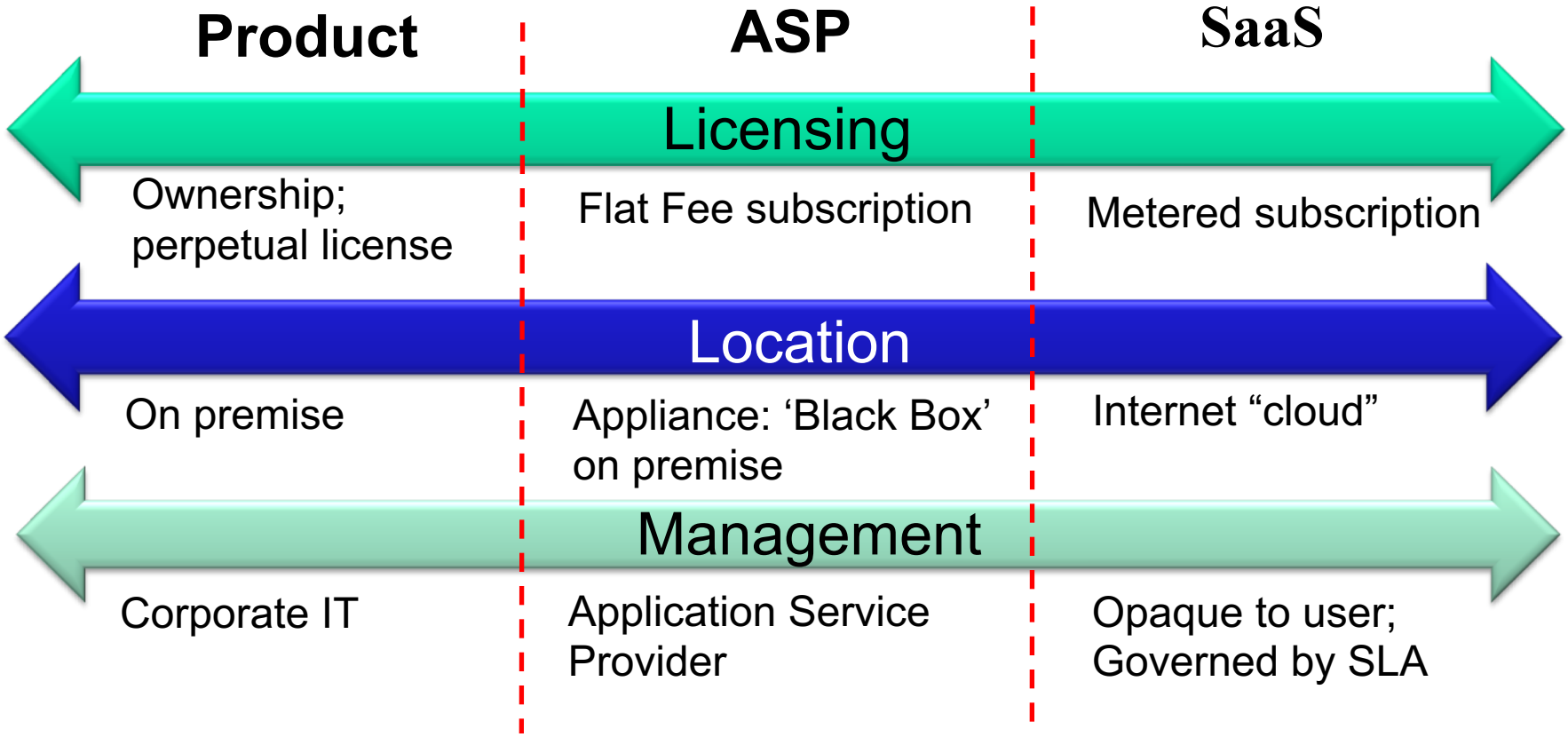
Two major categories:

- **Line-of-Business Services:** large customisable business solutions aimed at facilitating core business processes. Usually charged by subscription. Examples: finance, supply-chain management, CRM.
- **Customer-oriented Services:** offered to general public, consumer oriented. Mostly provided at no cost and supported by advertising. E.g. Hotmail.

Main Features of SaaS - IDC

- Network-based access to, and management of, commercially available software;
- Activities that are managed from central locations rather than at each customer's site, enabling customers to access applications remotely via the Web;
- Application delivery that typically is closer to a one-to-many model (single instance, multi-tenant architecture) than to a one-to-one model, including architecture, pricing, partnering, and management characteristics
- Centralized feature updating, which obviates the need for downloadable patches and upgrades.
- SaaS is often used in a larger network of communicating software – e.g. a plugin to a Platform as a Service (PaaS). SOA becomes critical foundation technology.

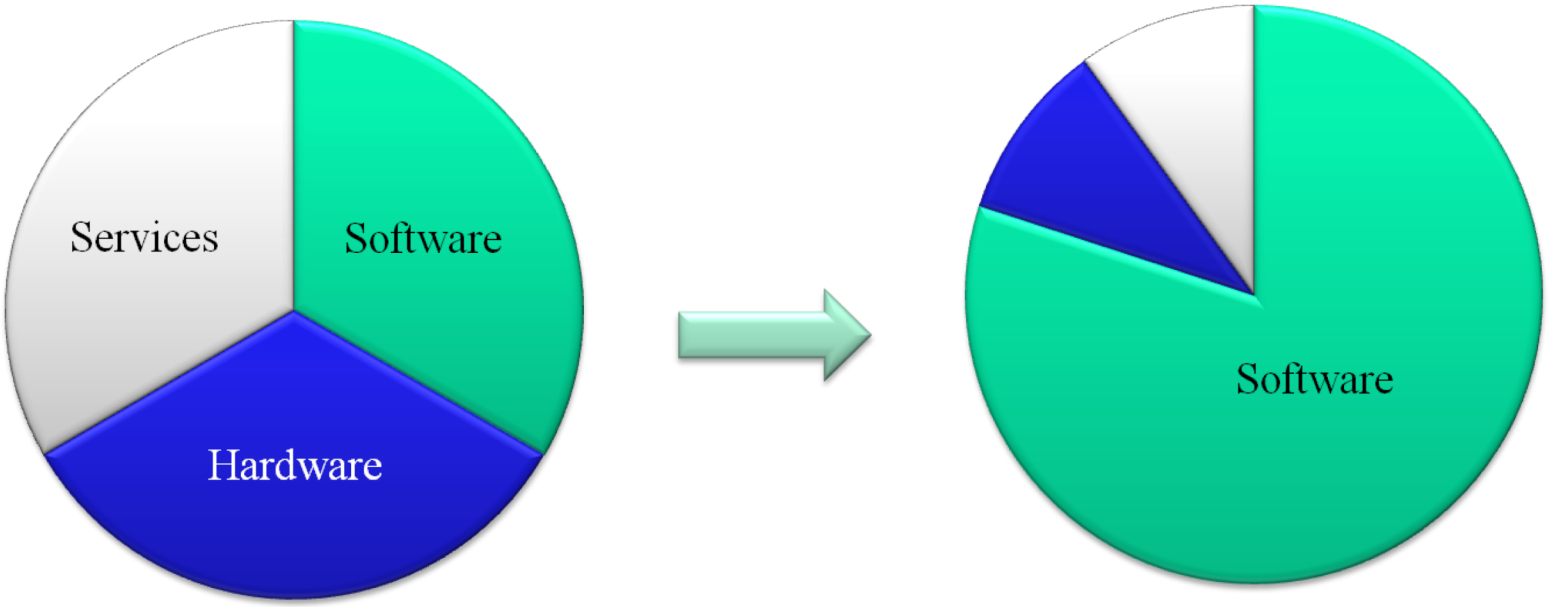
3 Differentiators – a Continuum



In 1990's "Shrink Wrapped" ASP model provided business applications over the Internet; this was a precursor to SaaS. These applications were built with single-tenant model; they did not share process and data with other applications, few economic benefits over locally installed product.

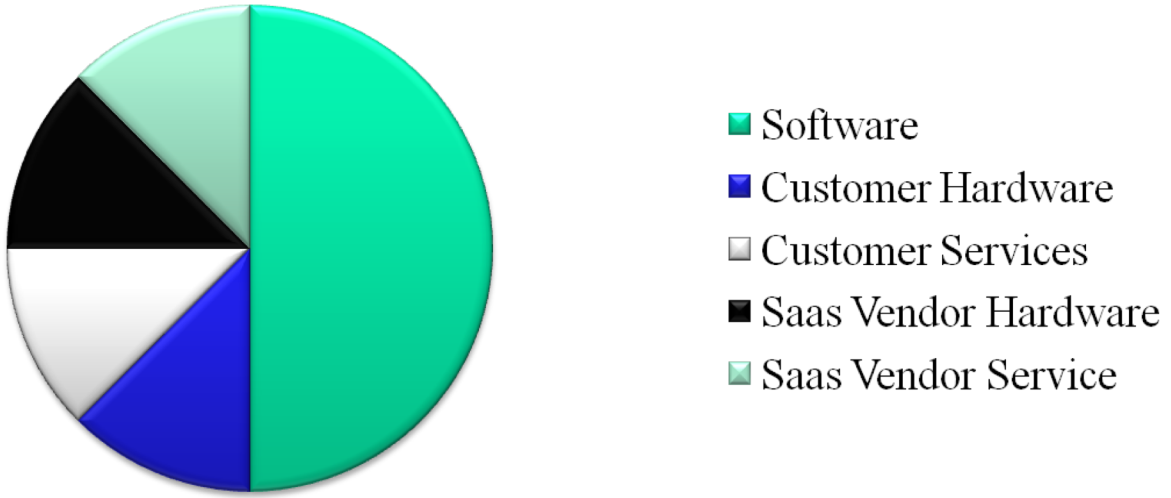
Changing the Customer's Business Model

- Shifting the "ownership" of software from customer to provider. Customer no longer "owns" a "product";
- Relocating responsibility for the technology infrastructure and management from customer to provider;
- Reducing the cost of services through specialisation and economies of scale;
- Targeting the "Long Tail"; by reducing the minimum cost at which software can be sold;
- Budget Changes: Traditionally Professional Services and Hardware take most of the IT budget. However with SaaS there is a major shift to Software.



Economies of Scale

- SaaS vendors service all customers in a consolidated site;
- Example: A traditional ‘product’ application may require 2 servers and 2 load balancers for redundancy. If the customer owns the software they have to buy all this hardware even if the application only uses ¼ of the server capacity.
- However, a SaaS vendor can use remaining ¾ of capacity for other customers. Multi–tenancy becomes a core competency, leading to a higher quality offering at a lower cost.
- Consider the considerable costs of high availability, disaster recovery, backup, maintenance, support etc.
- Even if SaaS vendor hardware and services costs are added in, the software budget is still greater for the customer than originally.

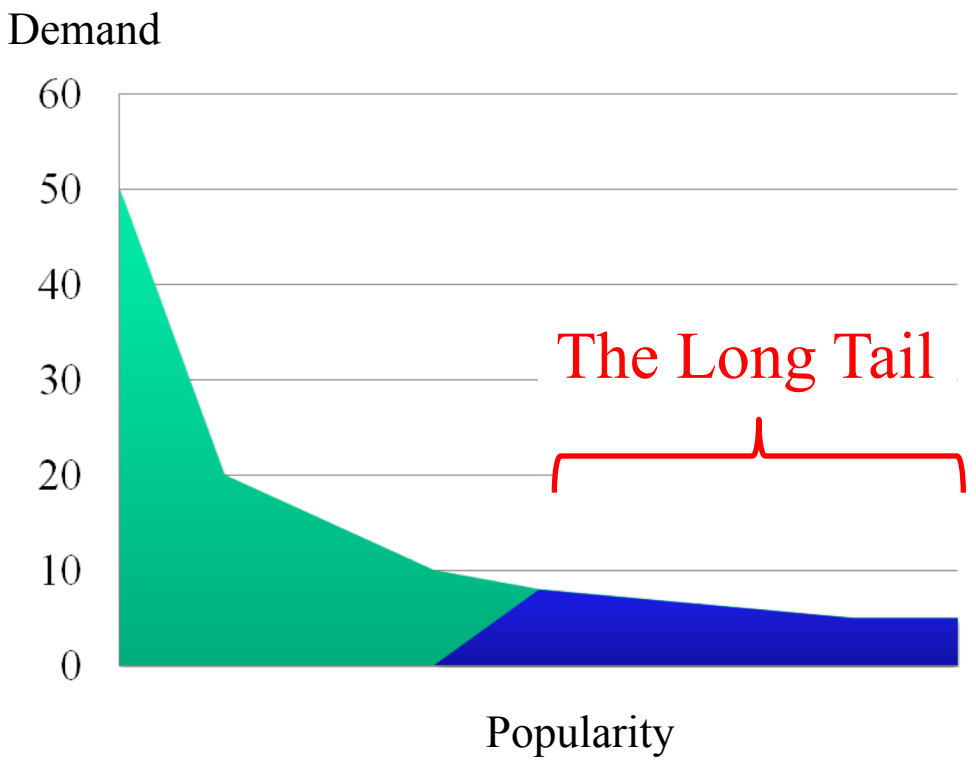


The Long Tail

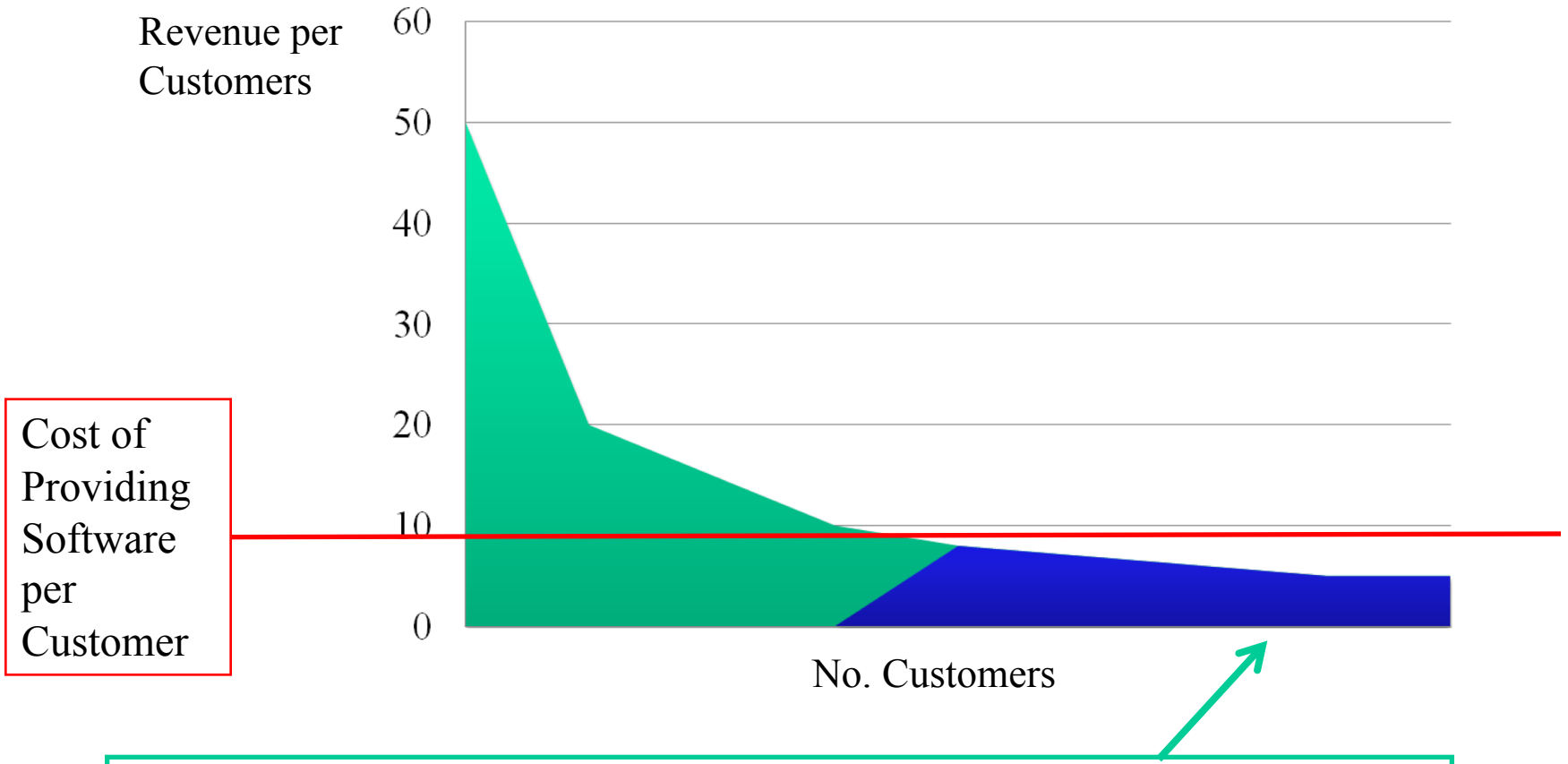
Chris Anderson popularized the idea of the "The Long Tail," in the October 2004 issue of Wired.

- Most retailers make money from stocking the best sellers;
- The remaining books languish in the "long tail' never seen on book store shelves;
- However, the majority of Amazons book sales come from outside its top 130,000.

- Software vendors face a similar market curve.
- Line of business software that requires on-site installation, configuring, support etc. comes at a price.
- This software has to be marketed at only those that can afford it and small businesses are left out.

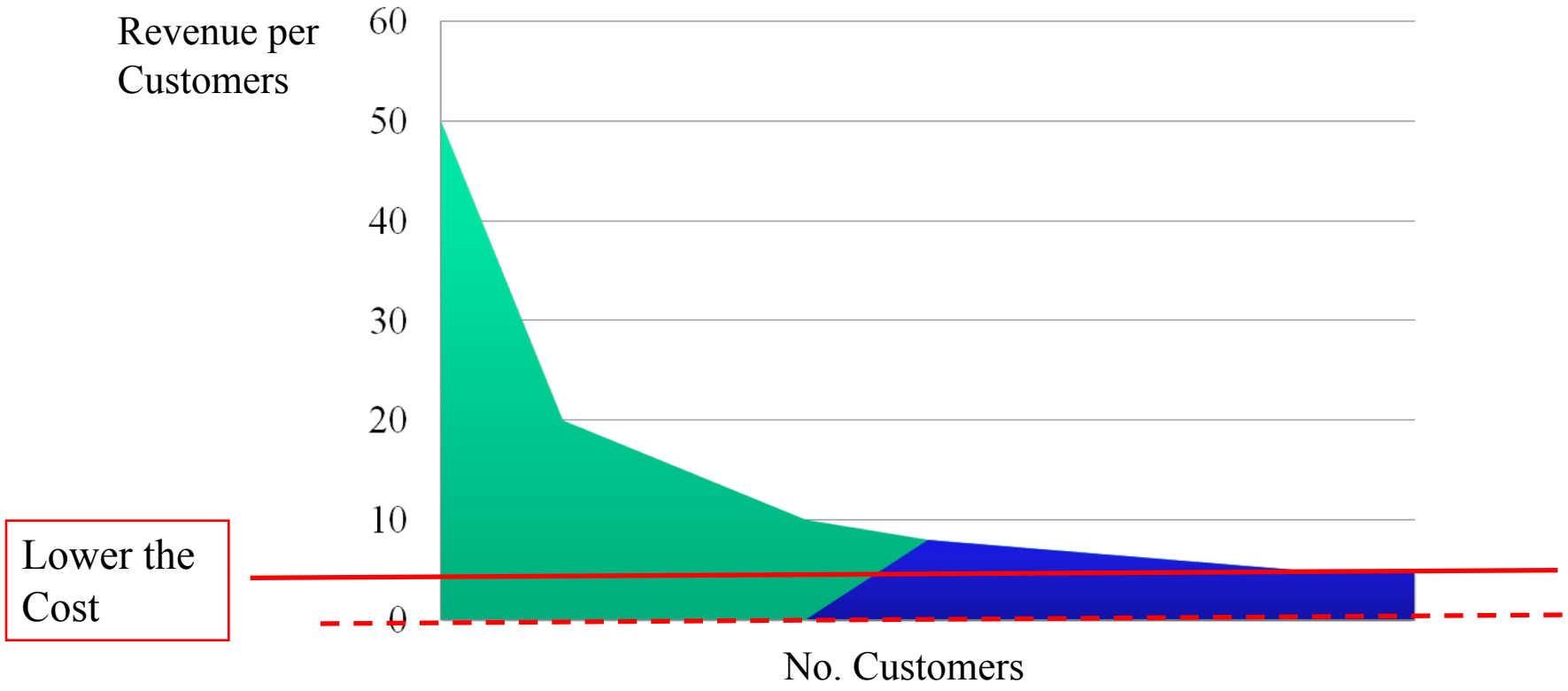


The Long Tail : Product Model



The “Long Tail” is a non-addressable market;
Costs are too high for small business to enter.

The Long Tail : Saas Model



- SaaS vendors can use their economies of scale to combine and centralise customer hardware and services' needs; to lower costs and provide higher quality;
- The "Long Tail" provides a new market.

In-house Application Build Issues

- Build cycles can take years, consume massive resource and are always late, unsatisfactory and over budget;
- Risks of software acquisition; massive upfront license costs;
- Integration is difficult and costly;
- Costs of maintaining and managing the infrastructure for the application, can be massive.
- Software and Hardware gets out of date quickly;
- Ongoing license and support fees;
- Large upfront CAPEX;
- Distributing updates to users and global accessibility is hard;
- Different applications, data and versions used across the globe;
- Mergers create headaches; more integration, more different applications to support, more users, more locations.
- Difficult and expensive to get into new markets;
- New SOA paradigm and Service Component model difficult to get into;

Benefits of SaaS to Customer

- SaaS provides same application across all sites, everyone is using the same, correct and latest version:
  - Updates are automatic;
  - Global accessibility;
- Systems are administered easily - No operational management;
- Scale easily via virtualisation;
- Companies can enter the market quickly and cheaply- low CAPEX
  - Cost spread across OPEX;
- Compliance (Sox, Hippa, SAS70 etc) managed by Vendor
  - Applications are logged and audited by vendor.
- Consolidated reporting;
- Management can focus on strategic activities;
- Try and buy approach – reduces risk on new ventures;
- SaaS vendors can provide better tools, better monitoring and better quality generally as they can use economies of scale to keep up to date and buy best of breed products.

Limitations of SaaS for Vendors

- Services must be well defined, that can achieve an economy of scale and the capacity to balance supply and demand. SaaS is not suitable for innovative or highly specialized niche systems.
- ‘Vendor lock-in', a lack of substitutability and second sourcing options creates a strategic weakness for any customer in terms of security, competition and pricing.
  - This situation is resolvable by the introduction of open sourced standards and the development of markets based upon such standards.
  - vendors counter the concerns over potential security and operational risk with the argument that the professionals operating SaaS applications may have much better security and redundancy tools available to them.
- Difficult for businesses that need extensive customization
  - is countered with the claim that many vendors have made progress with both customization and publication of their programming interfaces. It should be noted that customization will reduce substitutability and given that SaaS covers commodity-like activities, the strategic benefit of customization is highly dubious.
- The availability of open source applications, inexpensive hardware and low cost bandwidth combine to offer compelling economic reasons for businesses to operate their own software applications, particularly as open source solutions have become higher quality and easier to install.
- Users must be able to trust the provider of the service, particularly if the application stores the user's data.
- Large upfront cost for vendors, with return over time. GAAP forces the expenditure to be expensed upfront.

Architectural Features of SaaS

- Virtualised and/or Multi-Tenancy Models;
- High Scalability / Availability – often via Virtualisation;
- SOA – Service Composition/Orchestration;
- Re-Branding per Tenant;
- Partner Administration / Authorisation / Delegation
  - Each customer creates accounts for their users.
- Intelligent Transaction Management
  - Transactions are Tennant Specific;
- Intelligent Tenant Management:
  - Workflows are Tenant Specific and Configurable;
  - Data Customisable and Extensible, Meta-data Approach;
- Reporting / Billing per Tenant;
- System Qualities fully managed via SLA.
  - Hardware, network (data center) fully managed;

The Data Base Approaches

3 Alternative Tenant Models:

- Separate DB, Separate Schema
- Shared DB, Separate Schema
- Shared DB, Shared Schema

Shared initially more expensive to develop but cheaper overtime.

DB Approach Qualities

Approach	Security	Extensibility	Scalability
Separate DB	• Tenant Data Encryption, • Trusted Connection, • Secure Tables	Custom Columns	Single Tennant Scaleout
Shared DB	• Tenant Data Encryption, • Trusted Connection, • Secure Tables	Custom Columns	Tennant Based Horizontal Partitioning
Shared Schema	• Tenant Data Encryption, • Trusted Connection, • Tennant View Filter	• MetaData (OAV) extension tables • Preallocated generic fields	• Tennant Based Horizontal Partitioning • MetaData Scaleout

DB Complexities and Issues

- Scaling of base data V extended data will be different;
- May combine models e.g. Shared Data and Separate Tenant Data;
- More shared Data Less performance per Tennant;
- More Meta Data harder to index tables
- Application V Analytics DBMSs – users want reports;
- The more DB instances the higher probability of a failure;
- More tenants makes provisioning, back up restore more complex;
- Clustering and replication becomes complex, not handled well by open source DBMSs.

What about Layers Above DB

- Form and Data View Customisation;
- Business Logic Customisation;
- Report Customisation;
- Integration Customisation;

The above may require a Complex Meta Data and Dynamic approach too application design, architecture and delivery.

Service Delivery Platform

The complexities of SaaS means that to gain the economies of scale a platform is required to deliver it, that manages:

- Access Control and Security
- Auditing and Logging
- Order management
- Provisioning
- Management and SLA monitoring
- Metering and Billing
- Customer Support, Complaints and Ticketing

Then

- Application and Integration frameworks for above
- Customisation and configuration frameworks
- Device Delivery (Desktop, Mobile, Tablet)

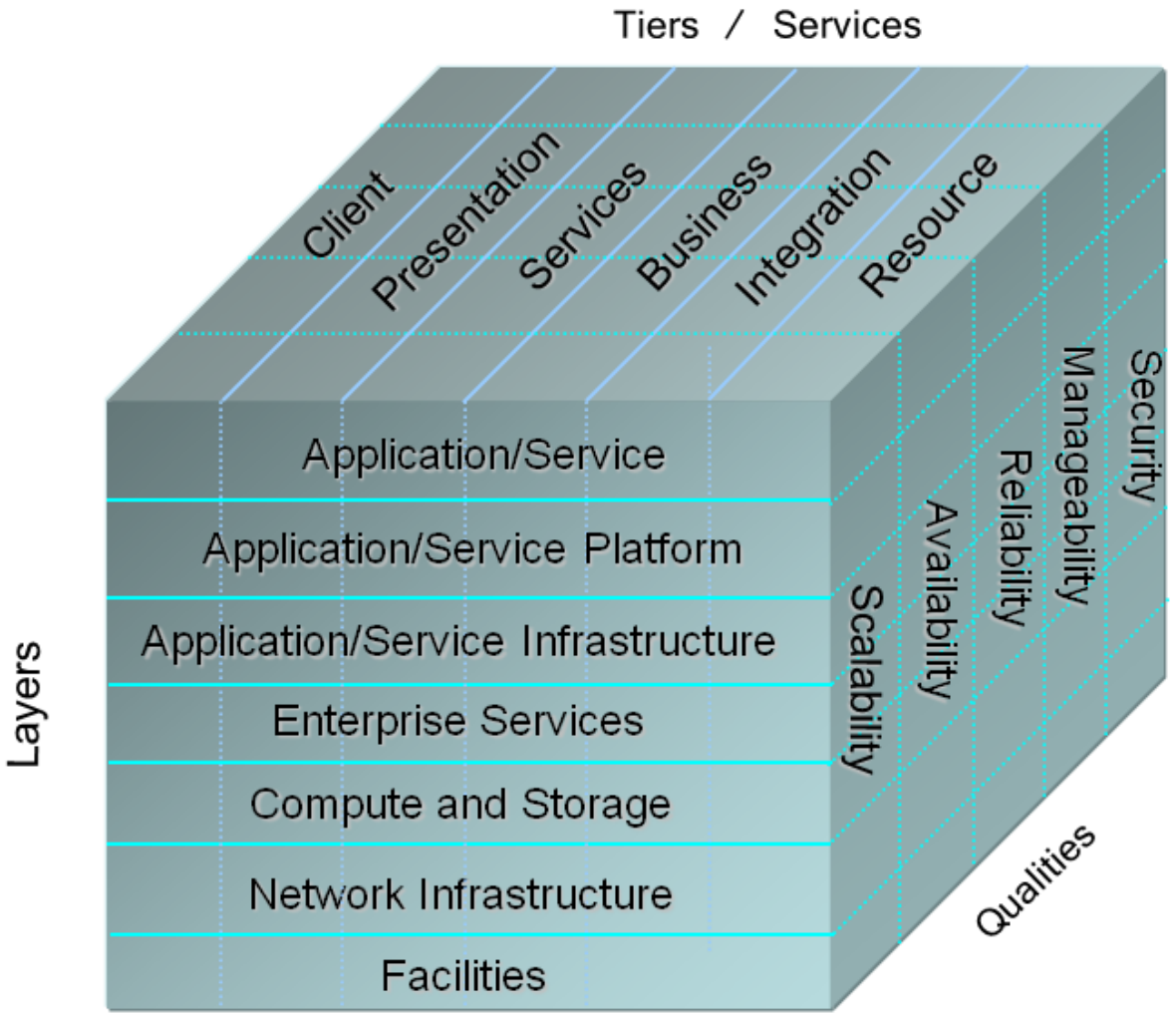
SDP, Who does What ?

Where do you draw the line ?

Tenant / Customer
Configures Stuff

Application SaaS
Product Owner
Builds Stuff

Owner Uses
Hosted Stuff



Integration Models

- Some customer just use the Web front ends – easy.
- Others require integration:
  - SOA, Model supports:
    - Push / pull, Publish Subscribe, Sync/Async
  - Customer may use an integration broker, Service Bus (ESB)
    - Open Source versions now mature.
  - Transaction management:
    - Managed by the service, Standards becoming prominent, e.g. WS-Atomic Transactions
  - Single Sign On – Identity management across applications, services.
  - Integrate using the Cloud, e.g. Adobe Cloud
  - Security and SSO: integrate to Customers Identity manager.
    - WS- Federation

What is the Cloud ?

- SalesForce.com, one of leaders in SaaS, say:
  - “Cloud is web based SaaS + web based, hosted dev platform”
- Google say:
  - “Google is the Cloud”
- Scott McNeally
  - “The Network is the Computer”
- Wikipedia
  - Cloud computing is a style of computing in which dynamically scalable and often virtualized resources are provided as a service over the Internet. Users need not have knowledge of, expertise in, or control over the technology infrastructure in the "cloud" that supports them.
- Both Business Services and Technical Services can be provided anywhere Internally or Externally

Layers of the Cloud ?

Top Layer: Software as a Service (SaaS)

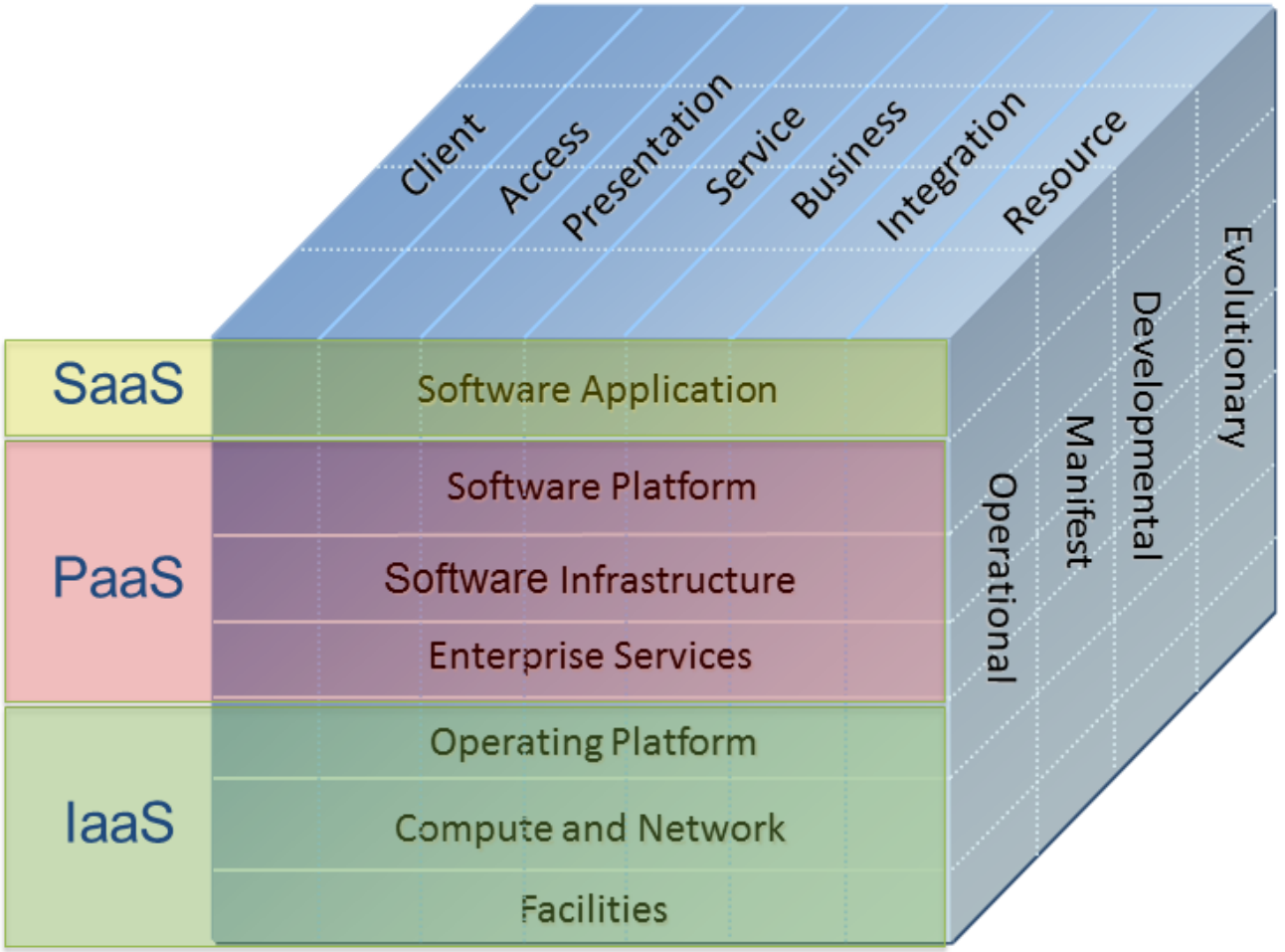
Middle Layer: Platform as a Service (PaaS)

- Deliver the Application Platform (Solution Stack); can include software dev and test stack.
- Also the concept of delivering a cost-effective cloud based workspace environment for the end-use
- Example: Amazon Web Service Xen Image, Google App Engine.

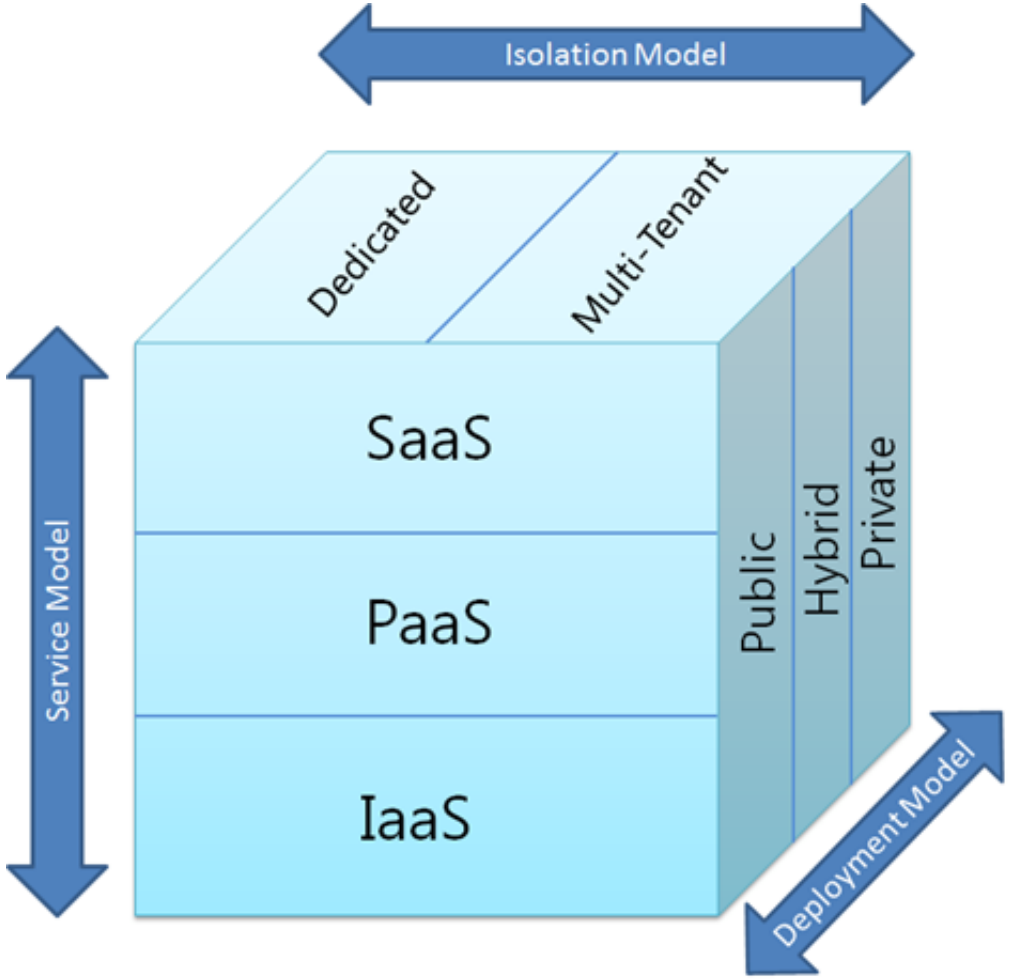
Bottom Layer: Infrastructure as a Service (IaaS):

- Deliver storage and compute capabilities as a standardised service to the enterprise (Server, Storage & Network Virtualisation)
- Provide the economies of Web companies to the enterprise.
- Example: Amazon Web Services, Ec2 and S3. SmugMug scale massively without any infrastructure.

Responsibility and Layers

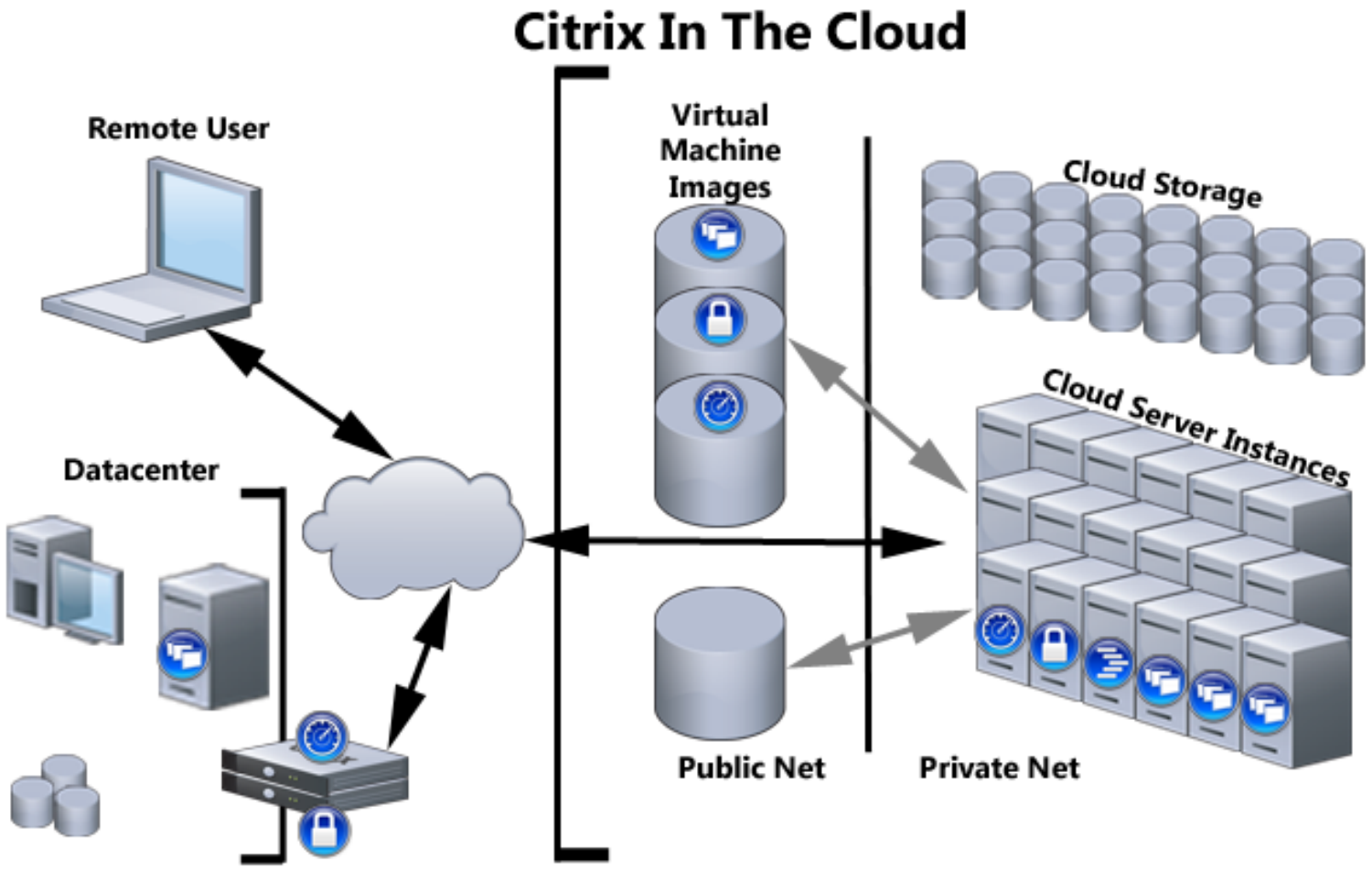


Categorizing the Cloud



From http://blogs.msdn.com/b/johnalioto/archive/2010/08/16/10050822.aspx

CITRIX Cloud Leader



Commercial Cloud Products

- Force.com is a cloud computing platform as a service offering from Salesforce (CRM, Social Computing etc…)
  - Apps are built using Apex a proprietary Java-like language and Visualforce an XML-like syntax for building user interfaces in HTML, AJAX or Flex.
- Amazon (many products e.g. payment)
  - EC3 : Elastic Computing Cloud – Scalable Xen servers;
  - S3 : Simple Storage Service – web service to store blobs;
  - AWS : Amazon Web Services;

- Google App Engine – Web PaaS
- Rackspace Open Cloud - Cloud Servers, Cloud Sites,
- Microsoft - Azure
- Citrix – Xen
- CastIron (IBM - Websphere)
- Others: Joyent, CloudSwitch, Verizon, RightScale, Heroku, JungleDisk, enStratus, BMC (BladeLogic, Atrium),

Open Source Products

- Eucalyptus
- Open Nebula (IaaS)
- jClouds
- OpenStack – Rackspace
- Lamp – Linux, Apache, MySQL, PHP
- Ubuntu Enterprise Cloud
- Redhat – private clouds
- WS02 Stratos (PaaS)
- Ecp - Enomaly's Elastic Computing Platform
- Other Projects: Chef, collectD, OpenQRM, Puppet, RabbitMQ, Zenoss, Bitnami, ControlTier, Cloud Foundry

Cloud Benefits

- For IaaS:
  - Move CAPEX to OPEX
  - Costs become more elastic;
  - Investment ties closer to usage allowing companies to ride the market
- For PaaS:
  - Automated Governance
  - No hardware concerns
- For SaaS:
  - Freedom from download, install, upgrade cycle;

Issues With Cloud

- Privacy and Security
- Compliance – FISMA, HIPAA, SOX, SAS-70, Data Protection Directive, PCI DSS,
- Making Changes to Applications or impact of platform changes to Application.
- Use of existing Management Tools
- Lock In.
- Open standards - OGF's Open Cloud Computing Interface.