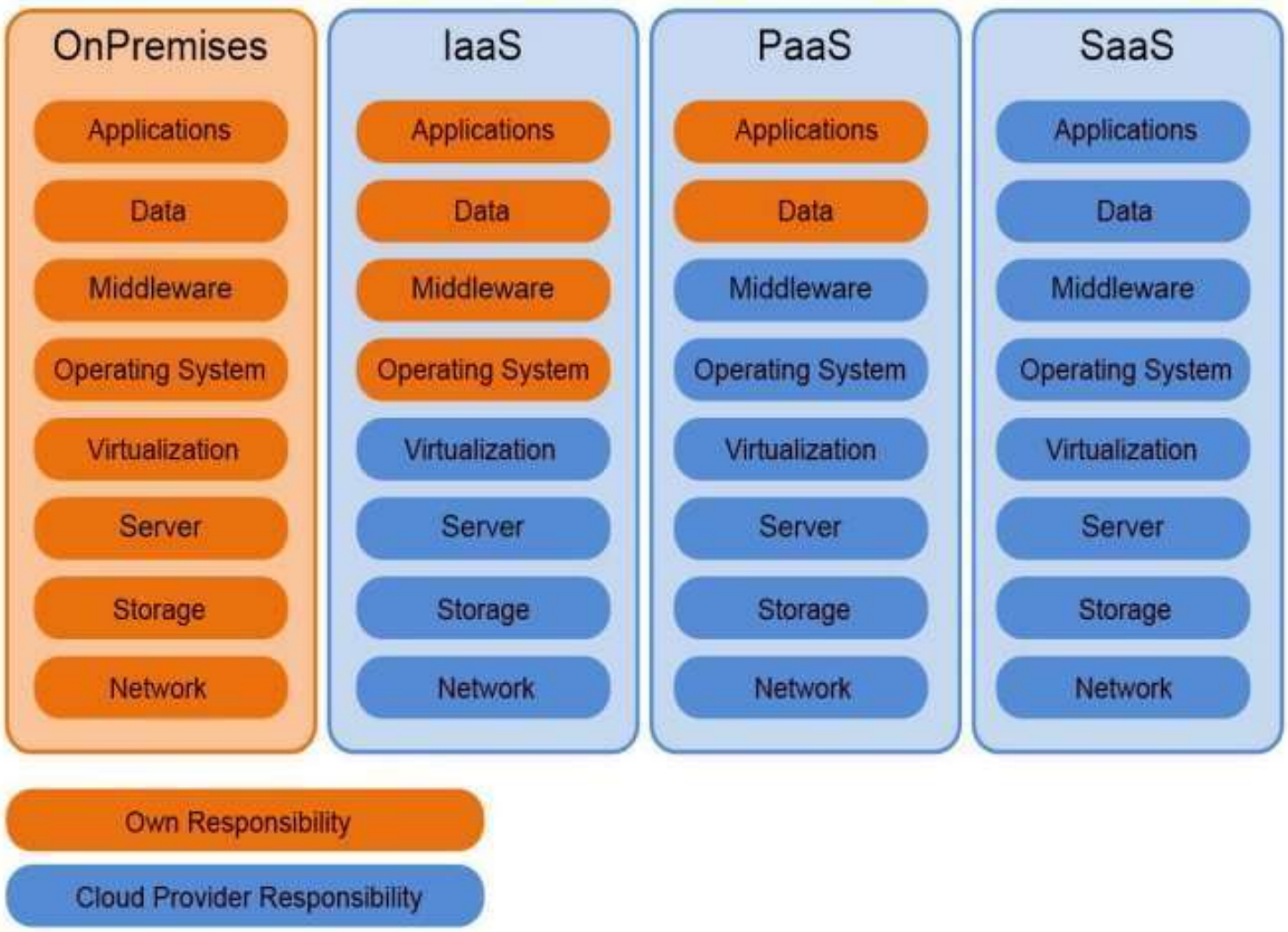
Cloud Computing

Session 10

PaaS and SaaS

BITS
Pilani

PaaS - Dependency on IaaS



Introduction to PaaS

- Platform as a Service, referred to as PaaS, is a category of cloud computing that provides a platform and environment to allow developers to build applications and services over the internet.
- Platform as a Service allows users to create software applications using tools supplied by the provider.
- PaaS services are hosted in the cloud and accessed by users simply via their web browser.
- PaaS services can consist of preconfigured features that customers can subscribe to; they can choose to include the features that meet their requirements while discarding those that do not.

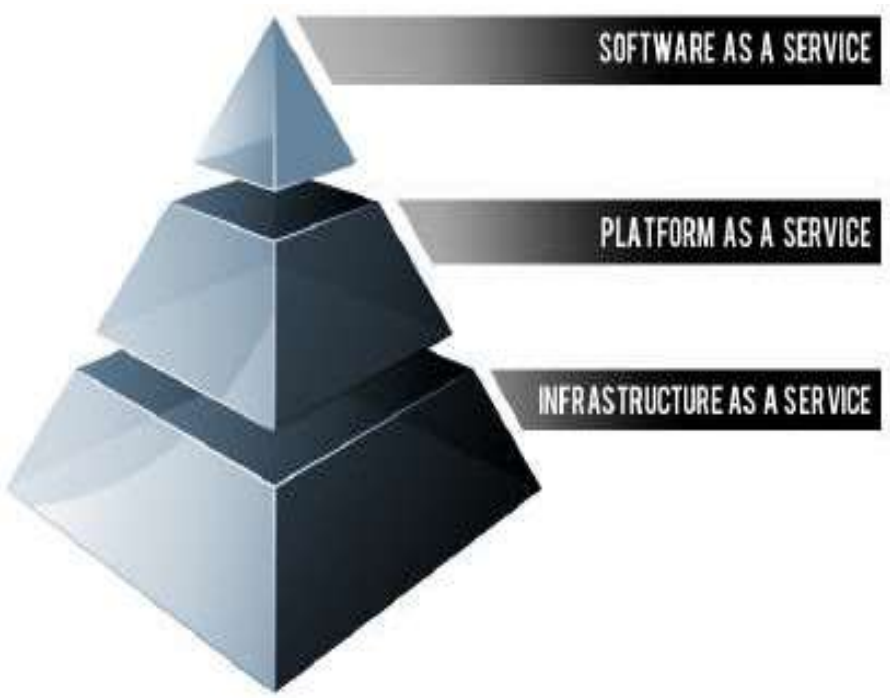
- Google App Engine, AWS Elastic Beanstalk

Building blocks of PaaS

PaaS providers can assist developers from the conception of their original ideas to the creation of applications, and through to testing and deployment.

Below are some of the features that can be included with a PaaS offering:

- Operating system
- Server-side scripting environment
- Database management system
- Server Software
- Support
- Storage
- Network access
- Tools for design and development
- Hosting



Benefits of PaaS

- Faster time to market
- Affordable access to a wider variety of resources
- More freedom to experiment, with less risk
- Easy, cost-effective scalability
- Greater flexibility for development teams
- Lower costs overall

How PaaS works?

In general, PaaS solutions have three main parts:

- Cloud infrastructure including virtual machines (VMs), operating system software, storage, networking, firewalls
- Software for building, deploying and managing applications
- A graphic user interface, or GUI, where development or DevOps teams can do all their work throughout the entire application lifecycle

Use cases for PaaS

- API development and management
- Internet of Things (IoT)
- Agile development and DevOps
- Cloud migration and cloud-native development
- Hybrid cloud strategy

Purpose-built PaaS types

- AIPaaS (PaaS for Artificial Intelligence) - lets development teams build artificial intelligence (AI) applications without the often prohibitive expense of purchasing, managing and maintaining the significant computing power, storage capabilities and networking capacity these applications require

- iPaaS (integration platform as a service) - a cloud-hosted solution for integrating applications.

- cPaaS (communications platform as a service) - lets developers easily add voice (inbound and outbound calls), video (including teleconferencing) and messaging (text and social media) capabilities to applications, without investing in specialized communications hardware and software.

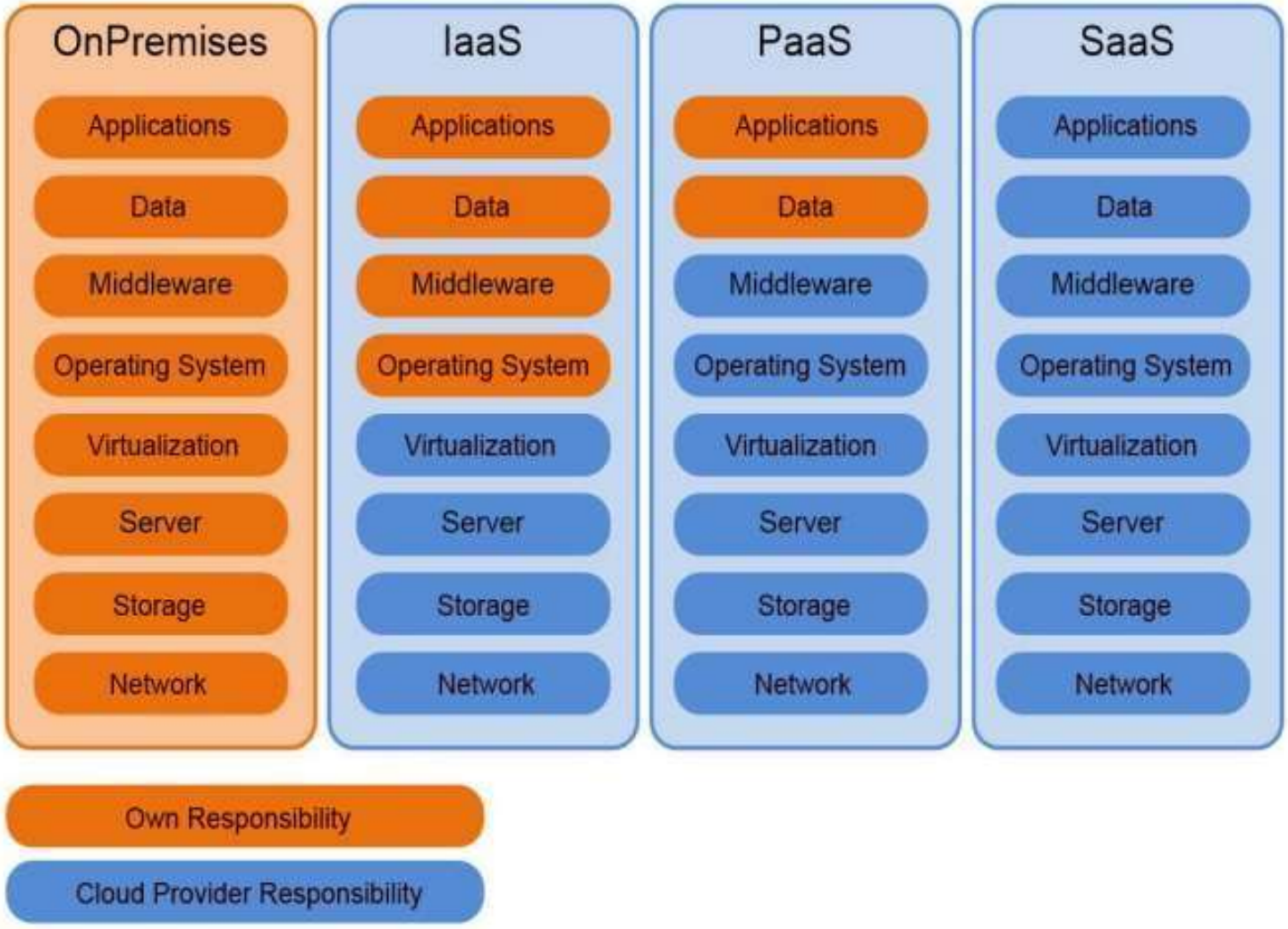
- mPaaS (mobile platform as a service) - simplifies application development for mobile devices.

SaaS

Software as a Service

BITS
Pilani

SaaS - Dependency on IaaS and PaaS



What is SaaS?

- Software as a service is a software distribution model in which applications are hosted by a vendor or service provider and made available to customers over a network, typically the Internet.

- Shortly, in the SaaS model software is deployed as a hosted service and accessed over the Internet, as opposed to "On Premise."

- Software delivered to home consumers, small business, medium and large business

– The traditional model of software distribution, in which software is purchased for and installed on personal computers, is sometimes referred to as software as a product.

Problems in traditional Model

- In the traditional model of software delivery, the customer acquires a perpetual license and assumes responsibility for managing the software.
- There is a high upfront cost associated with the purchase of the license, as well as the burden of implementation and ongoing maintenance.
- ROI is often delayed considerably, and, due to the rapid pace of technological change, expensive software solutions can quickly become obsolete.

Problems in traditional Model

Traditional Software

On-Demand Utility

Build Your Own

Plug In, Subscribe

Pay-per-Use

SaaS – How is it delivered

- The web as a platform is the center point.
- Network-based access to, and management of, commercially available (i.e., not custom) software 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

- Software delivered to home consumers, small business, medium and large business

– The traditional model of software distribution, in which software is purchased for and installed on personal computers, is sometimes referred to as software as a product.

SaaS – Architecture

- Run by
  - Internet technologies
  - The cost of a PC has been reduced significantly with more powerful computing but the cost of application software has not followed
  - Timely and expensive setup and maintenance costs
  - Licensing issues for business are contributing significantly to the use of illegal software and piracy.

SaaS Application Architecture

- Scalable
- Multitenant efficient
- Configurable

- **Scaling the application** - maximizing concurrency, and using application resources more efficiently
  - i.e. optimizing locking duration, statelessness, sharing pooled resources such as threads and network connections, caching reference data, and partitioning large databases.

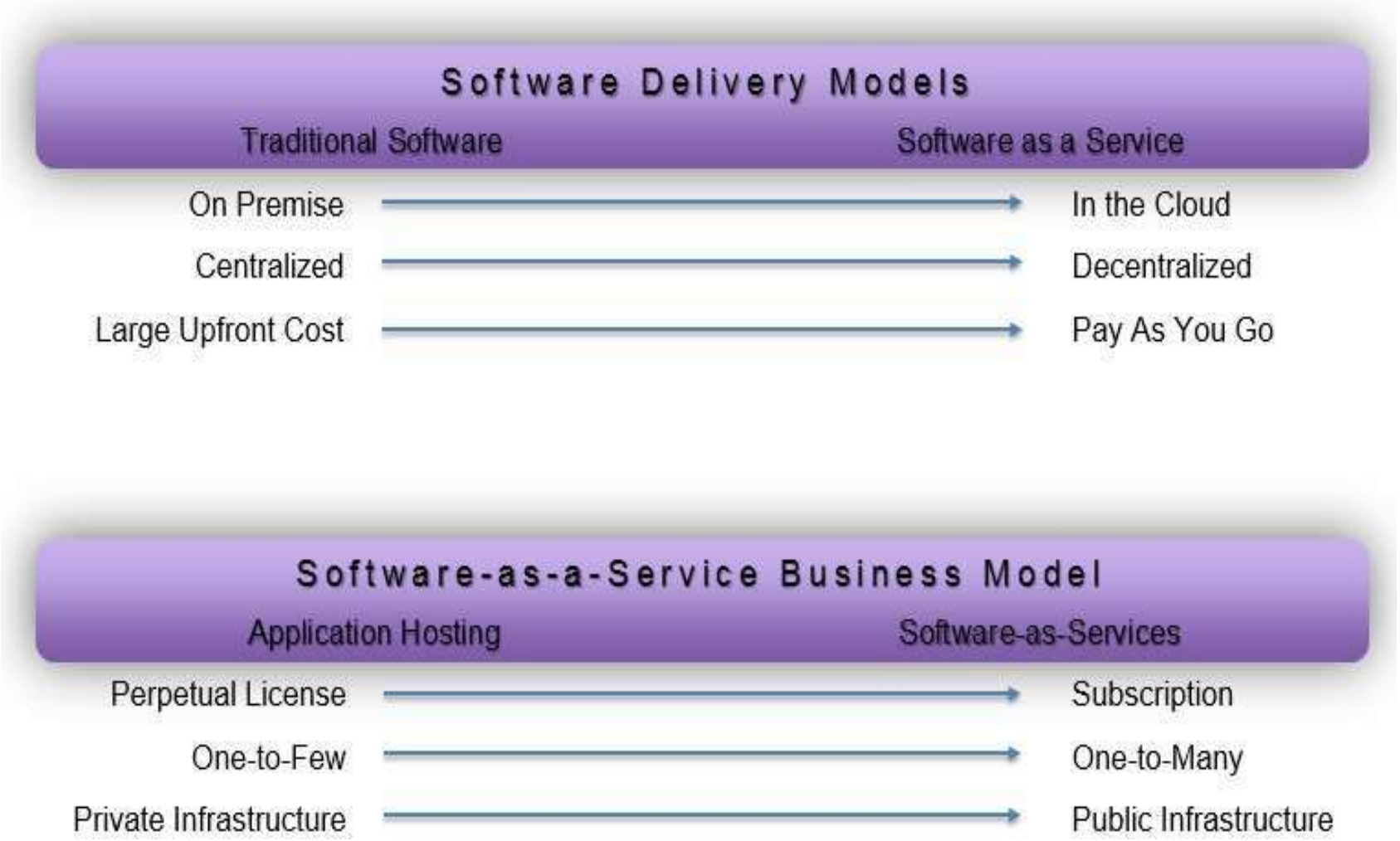
SaaS Application Architecture

- **Multi-tenancy** – important architectural shift from designing isolated, single-tenant applications
- One application instance must be able to accommodate users from multiple other companies at the same time
- All transparent to any of the users.
- This requires an architecture that maximizes the sharing of resources across tenants
- It should still be able to differentiate data belonging to different customers.

SaaS Application Architecture

- **Configurable** - a single application instance on a single server has to accommodate users from several different companies at once
- To customize the application for one customer will change the application for other customers as well.
- Traditionally customizing an application would mean code changes
- Each customer uses metadata to configure the way the application appears and behaves for its users.
- Customers configuring applications must be simple and easy without incurring extra development or operation costs

SaaS Models


Software Delivery Models
Traditional Software
Software as a Service
On Premise
In the Cloud
Centralized
Decentralized
Large Upfront Cost
Pay As You Go
Software-as-a-Service Business Model
Application Hosting
Software-as-Services
Perpetual License
Subscription
One-to-Few
One-to-Many
Private Infrastructure
Public Infrastructure

Business Model comparisons

Traditional packaged software

- Designed for customers to install, manage and maintain.
- Architect solutions to be run by an individual company in a dedicated instantiation of the software

Software as a service

- Designed from the outset up for delivery as Internet-based services
- Designed to run thousands of different customers on a single code

Business Model comparisons

Traditional packaged Software

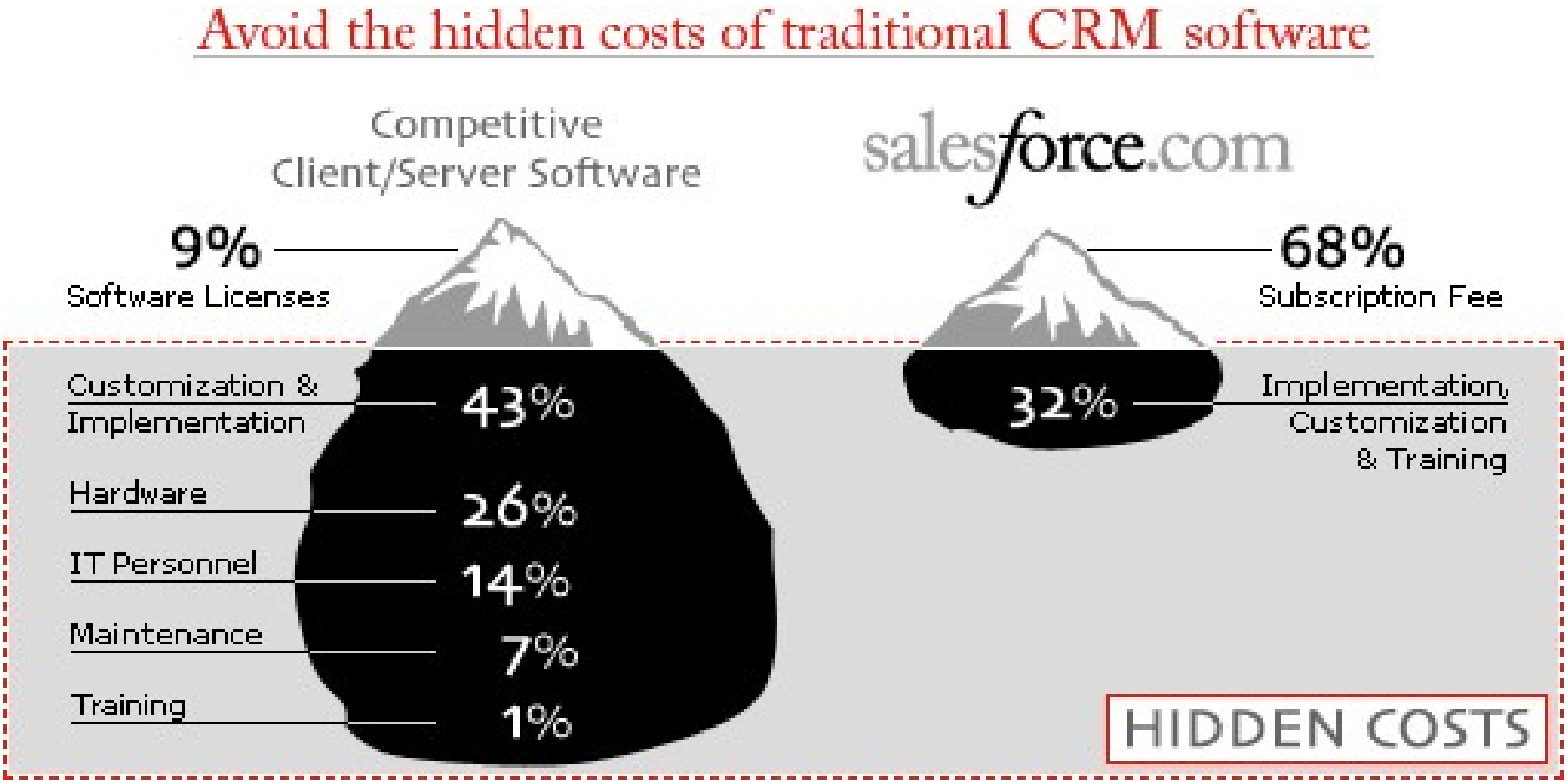
- Infrequent, major upgrades every 18-24 months, sold individually to each installed base customer.
- Version control
- Upgrade fee
- Streamlined, repeatable functionality via Web services, open APIs and standard connectors

Software as a service

- Frequent, "digestible" upgrades every 3-6 months to minimize customer disruption and enhance satisfaction.
- Fixing a problem for one customer fixes it for everyone
- May use open APIs and Web services to facilitate integration, but each customer must typically pay for one-off integration work.

Business Model comparisons

Hidden Cost



SaaS Advantages

Characteristics	Benefits
Network delivered access to commercially available software	**No local infrastructure or software to purchase or maintain** Applications & data are available anywhere with network connectivity
Application delivery is one-to-many model	Operating costs are reduced by managing infrastructure in central locations rather than at each customer's site
Built on optimized & robust platform	Improved availability and reliability
Customer pays for as much as they need when they need it	Lower TCO

Attributes	Software-as-a-Service (SaaS)	On-Premise
Alternate labels	On-Demand, Subscription Based, Hosted, Application Service Provider (ASP)	Installed, Hosted On-Premise
Purchase model	Lease, rent, subscription	Most commonly purchasing of licenses (ownership) with some lease and rent options
Maintenance and support	Typically included	For purchase option it would be based on the percentage of license fees
Security	Typically in a 3rd party highly secure data center	Responsibility of the costumer
Upgrades	Typically included	Included with maintenance
Data model	Shared (multi-tenant) or dedicated (single) instance depending on the vendor	Dedicated (single) instance on the customer's servers
Access	Typically all major web browsers	Typically all major web browsers
Initial investment	Low upfront cost since no hardware or infrastructure investment	Higher cost since typically purchasing licenses and hardware
3-5 year investment	Reoccurring fee	One-time fee
Legal implications	Hosted by 3rd party and lack of ownership. This needs to be clearly defined in a SLA	Less of an issue with the purchase option where costumer claims ownership
Implementation	Typically shorter since all is hosted by the vendor	Can be lengthier since it is installed and integrated with the existing infrastructure
Integration	Can be limited due to Web standards and protocols	Tends to be more flexible since it resides behind your firewall on your servers

SaaS User Benefits

- **Lower Cost of Ownership**
  - The software is paid when it is consumed, no large upfront cost for a software license Salesforce.com has a best-of-breed CRM system for $59.00 per user per month, with no upfront
  - Since no hardware infrastructure, installation, maintenance, and administration, budgeting is easy
  - The software is available immediately upon purchasing

- **Focus on Core Competency**
  - The IT saving on capital and effort allows the customer to remain focused on their core competency and utilize resources in more strategic areas.

SaaS User Benefits

- **Access Anywhere**
  - Users can use their applications and access their data anywhere they have an Internet connection and a computing device
  - This enhances the customer experience of the software and makes it easier for users to get work done fast

- **Freedom to Choose (or Better Software)**
  - The pay-as-you-go (PAYG) nature of SaaS enables users to select applications they wish to use and to stop using those that no longer meet their needs. Ultimately, this freedom leads to better software applications because vendors must be receptive to customer needs and wants.

SaaS User Benefits

- **New Application Types**
  - Since the barrier to use the software for the first time is low, it is now feasible to develop applications that may have an occasional use model. This would be impossible in the perpetual license model. If a high upfront cost were required the number of participants would be much smaller.
- **Faster Product Cycles**
  - Product releases are much more frequent, but contain fewer new features than the typical releases in the perpetual license model because the developer know the environment the software needs to run
  - This new process gets bug fixes out faster and allows users to digest new features in smaller bites, which ultimately makes the users more productive than they were under the previous model.

SaaS Vendor Benefits

- **Increased Total Available Market**
  - Lower upfront costs and reduced infrastructure capital translate into a much larger available market for the software vendor, because users that previously could not afford the software license or lacked the skill to support the necessary infrastructure are potential customers.
  - A related benefit is that the decision maker for the purchase of a SaaS application will be at a department level rather than the enterprise level that is typical for the perpetual license model. This results in shorter sales cycles.

SaaS Vendor Benefits

- **EnhancedCompetitive Differentiation**
  - The ability to deliver applications via the SaaS model enhances a software company's competitive differentiation. It also creates opportunities for new companies to compete effectively with larger vendors.
  - On the other hand, software companies will face ever-increasing pressure from their competitors to move to the SaaS model.
  - Those who lag behind will find it difficult to catch up as the software industry continues to rapidly evolve.

SaaS Vendor Benefits

Lower Development Costs & Quicker Time-to-Market

- The main saving is at testing (35%).
  - Small and frequent releases – less to test
  - Application is developed to be deployed on a specific hardware infrastructure, far less number of possible environment – less to test.
  - This, in turn, provides the software developer with overall lower development costs and quicker time-to-market.

Effective Low Cost Marketing

- Between 1995 and today, buyers' habits shifted from an outbound world driven by field sales and print advertising to an inbound world driven by Internet search.

SaaS Vendor Benefits

- **Predictable MRR Revenue**
  - Traditionally, software companies rely on one major release every 12-18 months to fuel a revenue stream from the sale of upgrades (long tail theory).
  - In the SaaS model the revenue is typically in the form of Monthly Recurring Revenue (MRR)
- **Improved Customer Relationships**
  - SaaS contributes to improved relationships between vendors and customers.
- **Protecting of IP**
  - Difficult to obtain illegal copies
  - Price is low, making getting an illegal copies totally unnecessary

Applicability – Scenario 1

Single-User software application

- Organize personal information
- Run on users' own local computer
- Serve only one user at a time
- Inapplicable to Saas model
  - Data security issue
  - Network performance issue
- Example: Microsoft office suite

Applicability – Scenario 2

Infrastructure software

- Serve as the foundation for most other enterprise software application
- Inapplicable to Saas model
  - Installation locally is required
  - Form the basis to run other application
- Example: Window XP, Oracle database

Applicability – Scenario 3

Embedded Software

- Software component for embedded system
- Support the functionality of the hardware device
- Inapplicable to Saas model
  - Embedded software and hardware is combined together and is inseparable
- Example: software embedded in ATM machines, cell phones, routers, medical equipment, etc

Applicability – Scenario 4

Enterprise Software Application

- Perform business functions
- Organize internal and external information
- Share data among internal and external users
- The most standard type of software applicable to Saas model
- Example: Saleforce.com CRM application, Siebel On-demand application