Fundamentals of Web Programming [a]

Software As A Service

Teodor Rus

rus@cs.uiowa.edu

The University of Iowa, Department of Computer Science

The Concept

Software as a service (SaaS) is a model of software deployment whereby a provider licenses an application to customers for use as a service on demand.

Approach:

- SaaS software vendors may host the application on their own web servers or download the application to the consumer devices, disabling it after use or after the on-demand contract expires.
- The on-demand function may be handled internally to share licenses within a firm or by a third-party application service provider (ASP) sharing licenses between firms.

Goal

The sharing of end-user licenses and on-demand use may reduce investment in server hardware or the shift of server use to SaaS suppliers of applications file services.

References

1. Finch, Curt (2006-01-02). The Benefits of the Software-as-a-Service Model.

   `http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=107276.`

2. Bennett, Keith; et al. (December 2000). "Service-based software" (PDF).

   `http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=896702`

3. SIIA (2001-02). Software as a Service: Strategic Backgrounder. `http://www.siia.net/estore/ssb-01.pdf`.

4. Archive.org crawled site in 1999. `http://web.archive.org/web/19991007185122/http://siteeasy.com/`

5. Traudt, Erin; Amy Konary (June 2005). "2005 Software as a Service Taxonomy and Research Guide". IDC. pp. 7.

More References

6. "Architecture strategies for catching the long tail". April 2006. Retrieved 2008-05-24.
7. Wainewright, Phil (October 2007). "Workstream prefers virtualization to multi-tenancy". Retrieved 2008-05-24.
8. Chong, Fred (October 2006). "Multi-tenancy and Virtualization". Retrieved 2008-05-24.
9. Schuller, Sinclair (March 2007). "Repealing the SaaS Tax". Retrieved 2008-05-24.
10. The Overlapping Worlds of SaaS and SOA
11. Gartner Survey Shows Many Users are Underwhelmed by Their Experiences of SaaS, Gartner.com, 2009-07-08. Retrieved 2009-09-16
12. "SaaS 2.0: Saugatuck Study Shows Rapid SaaS Evolution to Business Platforms". April 2006. Retrieved 2006-09-01.

History

The concept of "software as a service" started to circulate before 1999.[1]

Some breaking points:

- In December 2000, Bennett et al. noted the term as "beginning to gain acceptance in the marketplace".[2]
- The acronym "SaaS" was allegedly coined in the white paper called "Strategic Backgrounder: Software as a Service", which was published in February 2001 by the Software & Information Industry's (SIIA) eBusiness Divisions

Philosophy

SaaS does not concern computer user! Rather:

- SaaS is used by software professionals and business associates with the meaning of business software;
- It is typically thought of as a low-cost way for businesses to obtain rights to use software as needed versus licensing all devices with all applications.
- On-demand licensing enables the benefits of commercially licensed use without the associated complexity and potential high initial cost of equipping every device with the applications that are only used when needed.

Fact

Virtually all software fits the SaaS model.

Example:

- A licensed copy of a word processor had to reside on the machine to create a document.
- The equipped program has no intrinsic value loaded on a computer that is turned off for the night.
- Remote administration software attempts to resolve this issue through sharing CPU control instead of licensing on demand.
- SaaS achieves efficiencies by enabling the on demand licensing and management of the information and output, independent of the hardware location.

SaaS Applications

SaaS applications were developed specifically to leverage web technologies such as the browser, thereby making them web-native.

- The data design and architecture of SaaS applications are specifically built with a 'multi-tenant' backend, thus enabling multiple customers or users to access a shared data model;
- SaaS providers leverage enormous economies of scale in deployment, management, and support throughout the Software Development LifeCycle.

Characteristics

Main characteristics of SaaS software includes:

1. Network-based access to, and management of, commercially available software;
2. Activities managed from central locations rather than at each customer's site, enabling remote access via the Web;
3. Application delivery typically closer to a one-to-many model than to a one-to-one model, including architecture, pricing, partnering, and management characteristics;
4. Centralized feature updating, which obviates the need for end-users to download patches and upgrades;
5. Frequent integration into a larger network of communicating software (either as part of a mashup or as a plugin to a platform as a service).

Pricing

Providers of SaaS generally price applications on a per-user basis with a relatively small minimum number of users and often with additional fees for extra bandwidth and storage.

Consequences

- SaaS revenue streams to the vendor and therefore are lower initially than traditional software license fees.
- Recurring nature of SaaS use is viewed as more predictable, and much like maintenance fees for licensed software.
- In addition, SaaS software has these additional benefits:
  1. More feature requests from users since there is frequently no marginal cost for requesting new features;
  2. Faster releases of new features since the entire community of users benefits from new functionality;
  3. The embodiment of recognized best practices (since the community of users drives the software publisher to support best practice).

Implementation

SaaS architectures can generally be classified as being at one of four "maturity levels" whose key attributes are:

1. configurability,
2. multi-tenant efficiency,
3. scalability.

Each level is distinguished from the previous level by the addition of one of those three attributes.

Level 1

Ad-Hoc/Custom, characterized by:

- Each customer has its own customized version of the hosted application and runs its own instance of the application on the host's servers.
- Migrating a traditional non-networked or client-server application to this level of SaaS typically requires the least development effort and reduces operating costs by consolidating server hardware and administrations.

Level 2

Configurable, characterized by:

- Greater program flexibility through configurable metadata, so that many customers can use separate instances of the same application code.
- This allows the vendor to meet the different needs of each customer through detailed configuration options, while simplifying maintenance and updating of a common code base.

Level 3

Configurable, Multi-Tenant-Efficient, characterized by:

- A single program instance serves all customers.
- This approach enables more efficient use of server resources without any apparent difference to the end user, but ultimately comes up against limits in scalability.

Level 4

Scalable, Configurable, Multitenant-Efficient, characterized by:

- The fourth and final SaaS maturity level adds scalability through a multitier architecture supporting a load-balanced farm of identical application instances, running on a variable number of servers.
- The provider can increase or decrease the system's capacity to match demand by adding or removing servers, without the need for any further alteration of applications software architecture.

Virtualization

SaaS architectures may also use virtualization, either in addition to multi-tenancy, or in place of it.

Benefits:

- Virtualization can increase the system's capacity without additional programming. However, a considerable amount of programming may be required to construct a more efficient, multi-tenant application.
- Combining multi-tenancy and virtualization provides still greater flexibility to tune the system for optimal performance.
- In addition to full operating system-level virtualization, some virtualization techniques applied to SaaS include application virtualization and virtual appliances.

Components

SaaS applications may use various types of software components and frameworks.

Benefits:

- These tools can reduce the time-to-market and the cost of converting a traditional on-premise software product or building and deploying a new SaaS solution.
- **Examples:** include components for subscription management, grid computing software, web application framework.

Evolutionary Origin of SaaS

The origin of Web Services is the development of the distributed computer technology that support interoperability.

- A Web service is defined by the two HTTP methods: GET, POST and a vast collection of markup documents;
- Rather than deploying documents, one can deploy software components;
- Components can however be run on the Web server as remote services.
- Remote Procedure Call and the two technologies that followed DCOM (Microsoft) and CORBA (OMG) are best known examples.

Component Interoperation

Neither DCOM nor CORBA support universal component interoperability, which means:

"when a software needs a service it should be implicitly available on the Wei".

There are three roles required in order to use Web services:

1. Service providers (producers);
2. Service requesters (consumers);
3. Service registry.

Service Providers

A service provider must develop and deploy software that provide services.

- The service must have a standard description. The Web Service Description Language (WSDL) is the language designed for service description.
- WSDL is a W3C standard based on XML. Its specification is published on a Web server similar to other Web accessible documents.
- The description of service data (input and output) as well as the specific operations provided by a Web Service and the protocols for messages Web service can send/receive are written in WSDL.

Web Service Registry

The Universal Description, Discovery, and Integration (UDDI) is the registry created for Web Service Registration.

- UDDI provides methods for querying Web service registry to determine what specific services are available;
- UDDI has two kinds of clients: service providers and service requesters who query UDDI using WSDL queries;
- Service providers query the UDDI for service registry; service requesters query the UDDI for service use;
- UDDI registry respond with the protocol of how the providers/requesters may interact with the requested Web services.

SOAP

SOAP is an XML tag set that defines forms of massages and RPCs used by the Web services registered on the UDDI.

- SOAP was originally an acronym for Standard (Simple) Object Access Protocol, designed to describe data objects;
- The root element of a SOAP document in Envelope, so SOAP documents are also called envelops;
- The body of a SOAP message is either a request (which is a RPC) or a response (which contains values returned from the service;
- SOAP messages are sent using HTTP POST method.

Service Consumers

Web service consumers are clients of services which can be: Web applications, non-Web-applications, or other Web-services.

Web service client architecture:

- Web service client includes a proxy on the client machines which is a local substitute for the remote Web service;
- The client can call the methods of the remote service, so the calls are received by the proxy;
- Special tools are designed for this purpose. See Cloud computing and ASP.NET.

Another View

Web services are business objects that concern computer use as a problem solving tool.

However:

- The Web service business participants are all computer experts.
- On the other hand, computer use as a problem solving tool conquers all aspects of human life.
- Since software tools supporting the development and use of Web services are designed by and are targeted to computer experts, the current Web service business excludes most of its potential customers from their participation.
- To show this anomaly we take a look at the three main bricks supporting the Web service business: WSDL, SOAP, and UDDI.