# SaaS, PaaS, IaaS, and More: A Legal Primer on Popular Software Models

In today's rapidly evolving digital landscape, software has become an integral part of businesses across industries. With the rise of cloud computing and other new technologies, a range of software models have emerged, each with its own unique benefits and drawbacks. For busy professionals, it can be difficult to navigate this complex landscape and understand which model(s) may be most suitable for their needs. This article aims to provide a breakdown of the six basic software models, ranging from traditional software to open source and freemium software, and outline their key characteristics, advantages, and disadvantages. Whether you're a business owner or IT professional, this guide will help you make informed decisions about which model(s) to adopt and how to use them effectively.

We understand that our readers have varying levels of available time and interest. Therefore, we have structured this article to provide valuable takeaways for all readers, regardless of the amount of time they have to spare.

## Basic Software Models

For busy professionals who need concise information, we provide below a brief overview of the six key software models.

### Traditional software model:

In this model, the software is purchased or licensed by a customer and installed on their own computer system. The customer owns and manages the hardware and infrastructure required to run the software. The software may be customized or modified by the customer to meet their specific needs, and updates or upgrades may be provided by the developer.

### Software as a Service (SaaS):

In this model, the software is hosted by the provider and accessed by the customer over the internet on a subscription basis. The customer does not own or manage the underlying infrastructure, which is instead managed and maintained by the provider. The software is usually offered on a pay-per-use or pay-per-subscription basis, and the provider is responsible for security, performance, and maintenance of the software. Updates and upgrades are typically rolled out automatically to customers, without the need for additional installations or configurations.

### Platform as a Service (PaaS):

In this model, the provider offers a platform that customers can use to develop, run, and manage their own applications. The platform includes the necessary infrastructure and tools such as operating systems, databases, and development frameworks. The provider manages the underlying infrastructure, and the customer is responsible for developing and maintaining their own applications. Billing is typically based on

usage, such as the amount of data storage or computing time used by the customer.

### Infrastructure as a Service (IaaS):

In this model, the provider offers computing infrastructure such as virtual machines, storage, and networking. The customer has control over the operating system, applications, and deployed content, and can use the infrastructure to develop, run, and manage their own applications. The provider is responsible for managing the underlying infrastructure, including hardware and network components, and the customer is responsible for configuring and maintaining their own applications. Billing is typically based on usage, such as the amount of data storage or computing time used by the customer.

### Open source software:

In this model, the software is made available under an open source license that allows for modification, use, and distribution of the code by anyone. Users are typically free to modify the software to meet their specific needs, and updates and improvements can be contributed back to the community. The software is typically available for free, and revenue may be generated through support, consulting, or value-added services.

### Freemium software:

In this model, the basic version of the software is offered for free, with additional features and functionality available for purchase or subscription. The free version is usually limited in terms of functionality or usage, and users may be encouraged to upgrade to a paid version to access additional features or remove restrictions. Revenue may also be generated through advertising or other means.

## Advantages and Disadvantages of Each Model in Brief

For readers with more time available, keep on reading – we explore the main advantages and disadvantages of each software model in greater depth. We do this from both a developer and customer perspective.

### Pros and cons of traditional software model

– Developer perspective: Developers have complete control over the software, including distribution, updates, and licensing. This model can provide a higher profit margin for developers.

– Customer perspective: Customers have more control over their data and the ability to use the software even without an internet connection. However, customers bear the burden of maintaining the software and ensuring it runs properly.

### Pros and cons of Software as a Service (SaaS)

– Developer perspective: Developers can provide updates and maintenance more easily with an application running on their own servers. The subscription-based model provides a steady stream of revenue for developers.

– Customer perspective: Customers don’t need to worry about maintaining or upgrading the software. They are typically charged on a recurring basis, which can add up over time. Some customers may have concerns about data security and privacy.

### Pros and cons of Platform as a Service (PaaS)

– Developer perspective: Developers can focus on building applications rather than managing infrastructure. The provider takes care of server maintenance, scalability, and security.

– Customer perspective: Customers can reduce their IT infrastructure costs and focus on developing their own applications. However, they may be limited by the features and services provided by the platform.

### Pros and cons of Infrastructure as a Service (IaaS)

– Developer perspective: Developers have complete control over the software stack, including the operating system, applications, and networking. This model provides flexibility in terms of scaling and customization.

– Customer perspective: Customers don't need to worry about managing their own hardware and infrastructure. They can also scale their resources up or down quickly as needed. However, this model requires more technical expertise and can be more expensive.

### Pros and cons of open source software

– Developer perspective: Developers can benefit from a large pool of contributors who can help improve the software. Open source software also allows developers to showcase their work and collaborate with other developers.

– Customer perspective: Customers can use and modify the software as they see fit, without the need to pay licensing fees. However, customers may not receive the same level of support as they would with proprietary software.

### Pros and cons of freemium software

– Developer perspective: Developers can attract a large user base by offering a free version of their software. They can then generate revenue through ads, in-app purchases, or subscriptions for premium features.

– Customer perspective: Customers can try out the basic version of the software for free, which can be beneficial for those on a tight budget. However, some may feel like they are missing out on important features and functionality that are only available in the premium version.

## More Detailed Description of Each Software Model

For readers with ample time and a deeper interest in software models, we provide a more comprehensive summary of each model, including how they work, key features, and licensing agreement main provisions.

### Traditional Software Model

The traditional software model is a type of software distribution where a software package is sold or licensed to a customer for installation on their own computer system. The customer then has complete control over the software and is responsible for maintaining it.

In this model, developers sell software licenses to users, typically under an end-user license agreement (EULA) or a software license agreement. These agreements outline the rights and restrictions of both the developer and the user in regards to the use, distribution, and modification of the software.

Some key provisions of a typical software license agreement may include:

– Grant of license: The license agreement usually specifies the scope and limitations of the license that the user is granted to use the software.

– Intellectual property rights: Typically, the software developer retains ownership of all intellectual property rights in the software, including copyrights, patents, trademarks, and trade secrets.

– Restrictions on use: The license agreement may specify certain restrictions on the use of the software, such as limiting the number of devices on which the software can be installed.

– Warranty and disclaimer of liability: The developer may disclaim any warranty of merchantability or fitness for a particular purpose, and limit their liability for damages arising out of the use of the software.

– Termination: The license agreement may specify when or under what conditions the license may be terminated, such as in the case of breach of the agreement or non-payment of license fees.

Examples of traditional software models include Microsoft Office, Adobe Creative Suite, and QuickBooks, among others. In this model, customers purchase a license for a specific version of the software and are responsible for installing and maintaining it themselves.

One advantage of the traditional software model from a developer perspective is greater control over the software and potentially higher profit margins than other models. From a customer perspective, the traditional software model allows greater control over data and the ability to use the software even without an internet connection. However, customers also bear the burden of maintaining the software and ensuring it runs properly.

Updates and support for traditional software models are typically handled separately from the initial purchase of the software license. Customers may need to purchase additional licenses or pay for upgrades to access the latest version of the software, and support may be offered on a paid basis or through community forums and knowledge bases provided by the developer.

Overall, the traditional software model can provide a reliable and familiar option for both developers and customers, but it is important to carefully review and understand the terms of any licensing agreements before making a purchase.

## Software as a Service (SaaS)

The SaaS model, or software as a service, is a type of software distribution where the software is hosted by the provider and accessed by the customer over the internet on a subscription basis. The customer does not own or manage the underlying infrastructure, but instead pays for access to the software on a recurring basis.

In the SaaS model, customers typically enter into a software as a service agreement (SaaS Agreement) with the provider. Some key provisions of a typical SaaS Agreement may include:

– Grant of license: The SaaS Agreement usually specifies the scope and limitations of the license that the user is granted to use the software.

– Service level agreement (SLA): In SaaS agreements, providers often offer an SLA that provides guarantees for uptime, performance, and availability of the software.

– Intellectual property rights: The SaaS provider retains ownership of all intellectual property rights in the software, including copyrights, patents, trademarks, and trade secrets.

– Data ownership and privacy: The SaaS Agreement may specify who owns the data that is inputted into the software, as well as who has access to that data and how it will be used. Customers may also have options for exporting their data if they choose to terminate their subscription.

– Subscription fees and payment terms: The SaaS provider will typically charge customers on a recurring basis, either monthly or annually. Payment terms, including late fees and termination for non-payment, may also be included in the agreement.

– Termination: The SaaS Agreement may specify when or under what conditions the subscription may be terminated, such as in the case of breach of the agreement or non-payment of subscription fees.

One advantage of the SaaS model from a developer perspective is that it provides a steady stream of revenue and allows for easier updates and maintenance of the application running on their own servers. This model can also provide greater flexibility in terms of scaling and accommodating multiple customers. From a customer perspective, there is no need to worry about maintaining or upgrading the software, and they can access the software from any location with an internet connection. However, some customers may have concerns around data security and privacy.

Examples of SaaS models include Salesforce, Slack, and Dropbox, among others. In this model, customers subscribe to the service and access it through a web browser or dedicated app. The provider is responsible for keeping the software up-to-date and maintaining the necessary infrastructure to support the service.

Overall, the SaaS model provides a convenient and flexible option for both developers and customers, but it is important to carefully review and understand the terms of any SaaS Agreement before signing up.

## Platform as a Service (PaaS)

The PaaS model, or platform as a service, is a type of software distribution where the provider offers a platform allowing customers to develop, run, and manage their own applications. The provider manages and maintains the underlying infrastructure, such as servers, databases, and operating systems.

In the PaaS model, customers typically enter into a platform as a service agreement (PaaS Agreement) with the provider. Some key provisions of a typical PaaS Agreement may include:

– Grant of license: The PaaS Agreement usually specifies the scope and limitations of the license that the user is granted to use the platform and develop their own applications.

– Service level agreement (SLA): Similar to SaaS agreements, PaaS agreements often include an SLA that provides guarantees for uptime, performance, and availability of the platform.

– Intellectual property rights: Generally, the PaaS provider retains ownership of all intellectual property rights related to the platform, but may allow customers to retain ownership of their own applications.

– Data ownership and privacy: The PaaS Agreement may specify who owns the data inputted into the application and how it may be used by the provider.

– Subscription fees and payment terms: As with SaaS, providers typically charge customers on a recurring basis, often based on usage or tiered plans. Payment terms and conditions may vary depending on the provider and customer needs.

– Termination: The PaaS Agreement might specify when or under what circumstances the subscription may be terminated, such as in the case of non-payment, breach of agreement, or misuse of the platform.

Examples of PaaS models include Google App Engine, Microsoft Azure, and Heroku, among others. In this model, customers can use tools provided by the platform to create and deploy their own applications, without having to worry about the underlying infrastructure.

From a developer perspective, the PaaS model reduces infrastructure costs and allows developers to focus on developing and deploying their applications. The platform provider takes care of server maintenance, scalability, and security. From a customer perspective, they benefit from reduced IT infrastructure costs and can focus on developing their own applications. However, they may be limited by the features and services provided by the platform.

Overall, the PaaS model provides a convenient and cost-effective option for both developers and customers, but it is important to carefully review and understand the terms of any PaaS Agreement before signing up.

## Infrastructure as a Service (IaaS)

The IaaS model, or infrastructure as a service, is a type of software distribution where the provider offers computing infrastructure – such as virtual machines, storage, and networking – on a pay-per-use basis. The customer has control over the operating system, applications, and deployed content, but is responsible for managing their own software and data.

In the IaaS model, customers typically enter into an infrastructure as a service agreement (IaaS Agreement) with the provider. Some key provisions of a typical IaaS Agreement may include:

– Grant of license: The IaaS Agreement usually specifies the scope and limitations of the license that the user is granted to use the underlying infrastructure.

– Service level agreement (SLA): Similar to SaaS and PaaS models, IaaS agreements often include an SLA that provides guarantees for uptime, performance, and availability of the infrastructure.

– Intellectual property rights: Generally, the IaaS provider retains ownership of intellectual property related to the underlying infrastructure, but the customer retains ownership of their own applications and data.

– Data ownership and privacy: The IaaS Agreement may specify who owns the data inputted into the infrastructure and how it may be used by the provider.

– Subscription fees and payment terms: Providers typically charge customers on a recurring basis based on usage, such as compute time, storage, and bandwidth. Payment terms and conditions may vary depending on the provider and customer needs.

– Termination: The IaaS Agreement may specify when or under what circumstances the subscription may be terminated, such as in the case of non-payment, breach of agreement, or misuse of the infrastructure.

Examples of IaaS models include Amazon Web Services, Microsoft Azure, and Google Cloud Platform, among others. In this model, customers can choose the specific infrastructure resources they need and provision them as necessary.

From a developer perspective, the IaaS model provides complete control over the infrastructure, including the ability to customize and optimize the operating system and other components. This model also provides flexibility in terms of scaling, as customers can adjust their infrastructure resources as needed. From a customer perspective, they benefit from reduced hardware costs and can focus on managing their own software and data. However, this model requires more technical expertise and can be more expensive compared to other models like SaaS or PaaS.

Overall, the IaaS model provides a highly customizable and scalable option for both developers and customers, but it is important to carefully review and understand the terms of any IaaS Agreement before signing up.

## Open source software

Open source software is made available under an open source license that allows for modification, use, and distribution of the code by anyone. The development community collaborates on the software to improve it, often resulting in a higher quality product.

In this model, there is typically no formal contract between developers and users as the software is offered freely and openly. Instead, open source projects generally use standard open source licenses such as the GNU General Public License (GNU GPL) or the Apache License.

Some key provisions of the GNU GPL license, which is one of the most widely used open source licenses, include:

– Grant of license: The license grants permission to use, modify and distribute the software.

– Copyleft: This provision requires that any changes to the software must also be made available under the same open source license.

– Intellectual property rights: The license restricts the use of proprietary components in the software.

– Warranties and disclaimer of liability: The license disclaims any warranty of merchantability or fitness for a particular purpose, and limits the developer's liability for damages arising out of the use of the software.

From a developer perspective, the open source model provides an opportunity to showcase their work, collaborate with other developers, and benefit from contributions from a larger pool of developers. This model can also help increase adoption of the software and build a community around it. From a customer perspective, open source software is typically free and provides greater flexibility in terms of modifying and customizing the software to fit their needs. However, customers may not receive the same level of support as they would with proprietary software, and there may not be as many features available in comparison with commercial software.

Examples of open-source software include Linux, Apache web server, and MySQL database, among others. In this model, anyone can contribute to the development of the software and use it for their own purposes.

Overall, the open source model provides a collaborative and community-driven option for both developers and customers, but it is important to carefully review and understand the terms of any open source license before using or contributing to the software.

## Freemium software

The Freemium software model is a type of software distribution where the basic version of the software is offered for free, while additional features or functionality are offered for a fee. Customers can upgrade to the premium version by paying a one-time fee or a recurring subscription.

In this model, customers typically enter into a freemium software agreement with the provider. Some key provisions of a typical Freemium Agreement may include:

– Grant of license: The agreement usually specifies the scope and limitations of the license that the user is granted to use the software.

– Intellectual property rights: The provider retains ownership of all intellectual property rights in the software, including copyrights, patents, trademarks, and trade secrets.

– Fees and payment terms: Providers may charge fees for premium features, or offer subscriptions based on usage or tiered plans.

– Data ownership and privacy: The agreement may specify who owns the data inputted into the software and how it may be used by the provider.

– Termination: The agreement may specify when or under what circumstances the subscription may be terminated, such as in the case of non-payment or misuse of the software.

From a developer perspective, the Freemium model allows for a large user base by offering a free version of their software. This, in turn, can lead to increased adoption, greater brand recognition, and an opportunity to upsell premium features or services. Developers can then generate revenue through ads, in-app purchases, or subscriptions for premium features. From a customer perspective, they can try out the basic version of the software for free, which can be beneficial for those on a tight budget. However, some may feel like they are missing out on important features and functionality that are only available in the premium version, and there may be concerns around data security and privacy.

Examples of freemium software models include Dropbox, LinkedIn and Spotify, among others.

Overall, the Freemium model provides a flexible and scalable option for both developers and customers, but it is important to carefully review and understand the terms of any Freemium Agreement before signing up. Additionally, customers should be aware of potential hidden costs associated with using the free version of the software, such as limited storage, advertisements, or restricted features.

# Conclusion

In summary, there are various software distribution models available for both developers and customers to choose from. The SaaS model offers convenience and accessibility, PaaS provides a platform for development and deployment, IaaS provides complete control over the infrastructure, open source software allows for collaboration and flexibility, and the Freemium model provides a flexible and scalable option for both developers and customers. Each model has its own advantages and disadvantages, and it is important for both developers and customers to carefully review and understand the terms of any agreement before selecting a particular software distribution model. Ultimately, the decision should be based on individual needs, budget, and technical expertise, as well as the specific goals and requirements of the software being distributed.

* * *

This material is for general information only and is not intended to provide legal advice. If you have any questions or would like to learn more about software distribution models, please do not hesitate to contact us at info@danilovpartners.com. You may also find of interest our article on open source licenses.