Optimizing Software Supply Chains

Technology companies must view product development as a software supply chain, identifying the complexities of multiple business models and introducing a framework to mitigate risk.

Executive Summary

The software supply chain is best described as interconnected business processes that enable software products to be developed, converted to SKUs, distributed and provisioned (i.e., licensing and entitlement) to users. Most companies have a unique software supply chain, depending on the business model; but at a very high level, the sequence of steps remains the same. Hardware and high-technology product manufacturers that develop software for their devices/components also manage distinct supply chains for their software and hardware products to meet specific customer demands.

Although the sequence of steps might appear straightforward, much deeper complexities are involved when exploring each stage of the supply chain. For instance, many software companies source components from different vendors during product development, which can cause product integrity challenges. The risk increases when suppliers do not have robust processes in place to ensure their code conforms with industry-defined standards.

These complexities might change due to different factors, such as a particular company’s product portfolio or business policies. The impact can be significant if not managed effectively; for instance, the software company may risk an impending lawsuit if a lack of software integrity leads to client systems being exposed to vulnerabilities. Hence, it is essential for software companies to continually identify complexities and risks, and have a robust framework in place to mitigate them.

Many software companies do not view their business in terms of a conventional supply chain, and they fail to realize that all their business activities can be mapped to a single continuous value chain that should be optimized to achieve their business objectives.

This white paper presents a holistic perspective on software supply chain execution. It also highlights key areas where most companies have opportunities to optimize their value chains.

Software Supply Chain Trends

A company’s supply chain is defined by its business model. The software product industry has seen a continuous evolution of business models, starting with embedded systems and traditional software with perpetual licenses, moving to subscription-based software licensing and, finally, evolving most recently to SaaS models. Each business model has its own unique set of characteristics. Specific strategic and operational imperatives of each of these models are summarized in Figure 1, next page.
• **Embedded or traditional software**: This model is focused on selling one-time perpetual licenses as a key source of revenue and earning additional revenue through support contracts, annual version releases and bug fix releases. Typically, such products are sold via multiple channels and have a relatively longer product release cycle.

• **Subscription model**: Here, the license defines the entitlement for specific modules/components of the software, and the focus is on renewing subscriptions as a key source of revenue. Such products are also sold via multiple channels and have multiple SKUs. Entitlement, metering and billing are the key support processes needed for such products.

• **Software as a service**: SaaS involves on-demand provisioning of software, with a focus on billing users only for the portion they use. The focus is on subscription renewals as a key source of revenue. Distribution of products is direct and online, and metering and billing are the only key support processes needed.

Certain business drivers are shared among these business models, while other drivers are unique to a particular model. For instance, both the subscription and SaaS models have subscription renewals as their main revenue driver, and pricing is usually subscription- or consumption-based. With subscription models, the most critical operational processes are entitlement management, metering and billing, while SaaS requires no entitlement management because the product is provisioned directly to the user.

Similarly, subscription models are similar to traditional perpetual license models in that both have indirect and channel-heavy product distribution conduits. Traditional licensing models, however, are based on one-time license fees, whereas subscription models rely heavily on renewals.

As a result of these business drivers, each model differs in terms of supply chain and business complexity. As depicted in Figure 1, a software subscription model has higher business and supply chain complexities compared with traditional software or embedded software models. The reason: It requires complex systems to manage subscription entitlement and renewal processes.

The SaaS model, on the other hand, has much lower supply chain complexity than the subscription model. This is true because it doesn’t require complex distribution networks, and it has a much higher business complexity due to the complex metering and pricing/billing mechanisms needed to ensure that each customer is charged based on usage only.

What’s noteworthy is the fact that no technology company is reliant on a single business model, and businesses typically blend different models to address the needs of various customer segments. For example, while Autodesk has made most of its leading product suites available on SaaS (e.g., Maya, Inventor, etc.), the company serves a large portion of its customer base (such as educational institutions, private research labs, etc.) via a conventional subscription model, which provides limited entitlement software. The company’s supply chain, therefore, must be robust enough to accommodate multiple business models.

**Business Models Differ in Supply Chain Complexity**

![Figure 1](image-url)
Anatomy of the Software Supply Chain

A software supply chain can be broadly defined as a chain of processes that enables products to be built from components or modules, some of which may be sourced from different vendors and then distributed via different channels to meet customer demand. Figure 2 offers a high-level representation of the software supply chain, highlighting the key imperatives of each stage. The key stages can be defined as:

• **Sourcing and building:** This entails sourcing different components/modules of the product from different teams or vendors and integrating them to build the product. The integrity and authenticity of the code is maintained by different control processes.

• **Enabling product SKUs:** This involves several activities: developing product SKUs by bundling different product components and setting pricing appropriately based on target customer segment; maintaining an exhaustive bill of materials and listing configurations that define product SKUs; and developing pricing models.

• **Provisioning and licensing:** This includes the distribution of product licenses to customers, managing entitlements, metering and billing, and providing customer support.

• **License end-of-term:** This covers the management of support processes for subscription renewals or terminating entitlement of a user.

Sourcing and Building

The sourcing and building stage deals with developing/coding the product from scratch and integrating different code components to develop an end product. Typically, product development involves sourcing different components/modules from different teams or vendors and building product SKUs. There could be many prospective sources of code, including developed in-house, contract sourcing (where some of the modules are developed by a vendor), open source (leveraging open source code and customizing it) and commercially available code for specific software/components.

When sourcing the code from various providers, it is imperative that the entire sourcing chain is free from any intentional or unintentional vulnerability. This will ensure that the final product not only performs all the planned functions but also protects end-users from financial or legal risk (e.g., using third-party or open source code might create royalty obligations of which the product team might be unaware).

Minimizing vulnerabilities necessitates robust software assurance processes and control measures to ensure all possible risks are mitigated (see Figure 3, next page). Among the key control measures that should be considered:

• **Policies:** Formulating governing policies with vendors via registration/agreements and then communicating these policies ensures code check processes are properly instituted.

Software Supply Chain: An Overview

![Software Supply Chain Diagram](image)

Figure 2

cognizant 20-20 insights
• **Training and certification:** Periodic training and (re)certification programs help employees stay abreast of strategies for reducing risk. For example, training employees on standard software check-in processes reduces or eliminates the risk of having unauthorized code in the product’s code base.

• **Control self-assessment:** Self-testing can be used for low-risk activities.

• **Automated control:** Automation of high-risk controls ensures greater compliance. For example, automation of “software watermarking” for third-party code to enable tracking of its usage minimizes the instances of unauthorized code.

• **Internal audit:** Third-party validation of control processes adds a new perspective to compliance and risk mitigation strategies.

Different business units or product portfolios within the same company may vary in their level of dependency on sources of code; because of this, they will require different software assurance strategies and control measures. For example, IP risk compliance is mainly dependent on three factors: third-party code content, size of the engineering group and utilization of products. Hence, a company with multiple business units will have multiple exposures to these factors and, therefore, will require different software assurances.

For instance, if a business unit has a low probability/likelihood of IP infringements due to these factors and if the impact is low, then that business unit can focus just on monitoring risk. However, a more stringent risk mitigation framework should be adopted if both the likelihood of IP infringements and the impact are high.

**Enabling Product SKUs**

This stage deals with the steps required to transform an end product into a market-ready product.

• **Defining software product SKUs:** Software is similar to any product that is categorized by SKUs. Some software can be purchased with a license for a specific number of machines, but the same software could be purchased at another time with a license for a specific number of users. Each of these is a unique instance that needs to be tracked separately as a SKU.

• **Source of product complexity:** Product innovation to attain market leadership, support local requirements and tailor solutions to customers’ needs leads to increased product complexity and a proliferation of SKUs. A robust and flexible bill of materials engine is needed to accommodate a large number of SKUs.

• **Measuring the cost of product complexity:**

  > **Analyze the contribution margin of a SKU:** Plotting the cumulative revenue or margin contribution of products in descending order helps identify those that represent the contribution majority.

**Governance Mitigates Rising Risk, Cost of Compliance**

![Diagram of Governance Mitigates Rising Risk, Cost of Compliance](image)
Item productivity analysis: Item productivity analysis helps limit the number of items and configurations in the supply chain that add little value to the portfolio.

- Addressing product complexity/SKU management. An effective strategy for addressing product complexity focuses on cultivating value-added complexity while reducing complexity that leads to inefficiency and lower profitability. Moreover, software providers must:
  - Understand the unique requirements of different customers and identify clusters of demand for customer segmentation, supply chain segmentation and cost-to-serve integration to profitably align supply chain capabilities with customer needs.
  - Enable the supply chain to more effectively deliver total customer solutions that include hardware, software and services.
  - Take supply chain considerations into account while establishing product design specifications.
  - Conduct reviews following product launch. Reviewing the success of a product launch and its impact on the supply chain helps organizations understand the performance of that product against the initial targets/forecasts. It will also lead to improvements in subsequent product introductions.
- Enhancing governance processes: Establishing centers of excellence and program governance bodies driven by business units reduces non-value-added complexity.

- Setting configuration, SKU or unique component reduction targets: Organizations need to understand the opportunity cost of producing low-profit, low-velocity SKUs and configurations, as well as the resulting improvement in product portfolio profitability if those products were eliminated.
- Defining business practice rules for SKU management: Rules might include “one in/one out;” caps on line variants per year by category; and revenue, volume and margin contribution thresholds.

Provisioning and Licensing

Once the product SKUs are defined and pricing is decided, the next stage deals with facilitating product distribution (i.e., provisioning the product). The main activities during software provisioning include:

- Software delivery: Making sure customers get their product. This can be done via the traditional physical shipment of software CDs or through electronic delivery, which is the current norm.
- Software updates: Ensuring customers get all the software updates as defined by their software entitlements. The mechanism of sending updates could be:
  - Pull-based: Users are provided with periodic electronic communications or alerts that they can access to update their software.
  - Push-based: Software vendors provide a means of updating software via the Internet.
- Licensing and activation: Activating customer licenses so they can use software for the intended purpose.
- Entitlement: Ensuring customers are able to get all the features they paid for.
- Customer support: Resolving customer issues related to software use.

Software publishers do not license their products in a standard way, which means no single method will cover all the possible permutations of license metrics. License complexity is increasing, particularly as hybrid license models emerge that encompass usage- and device-based licensing models. Entitlement is a key aspect of software provisioning, and with changing software licensing standards and models, software publishers must more closely scrutinize typical entitlement gaps.

Even though most software is distributed electronically, the provisioning processes still need to be managed. Entitlement engines help establish rules to ensure that software licenses are registered correctly; for example, licenses can have geographic or market restrictions based on price, usage, transfer, etc., for product support and upgrades.
Figure 4 illustrates how and why the entitlement process for subscription software is more complex than for other types of business models. Entitlement gaps exist even for installed software, which is why new systems geared toward license optimization have emerged to fill the gap.

To stay ahead of the competition and streamline the provisioning process, many software vendors are:

- Building cloud-based automated entitlement engines for SaaS, which is a vast improvement over manual entitlement service contracts used with traditional licensed software.
- Building and/or adopting new delivery models, such as e-delivery and hosted delivery.
- Introducing balanced scorecard metrics (e.g., customer support response, time to activate licenses) to better understand distribution dynamics.
- Providing proactive customer support, such as:
  - Push updates/information to customers and channel partners.
  - Self-help tools around license activation, deactivation and FAQs.

### License End-of-Term

The end-of-term for a license generally leads to either renewing the subscription entitlement, moving to a new product/pricing model or terminating the subscription altogether. For traditional on-premise software that has reached the end of extended maintenance, it could also mean upgrading to the latest version. Software providers generally institute proper software licensing and compliance to manage the end-of-term process by effectively performing:

- **Renewals/upgrade management:** Companies benefit from investing in automated solutions to track license renewal opportunities to minimize revenue leakage, while also cross- or upselling new products and services. SaaS companies must also ensure that customers who have chosen not to continue with the subscription are managed well and not billed incorrectly.

- **Usage tracking:** This is programmatically managed by including utilities in the software to track concurrency overage and software duplication.

- **Notifications and response management:** Software vendors need a mechanism to send automated notifications to customers for end-of-term action to renew or deactivate licenses. The solution must also help companies track customer responses for renewal, cancellation or deactivation.

To address these activities, companies must take a more graduated approach to licensing and construct policies based on customer size, product and geography. The bottom line is that a clear software license policy is essential for software companies to succeed.

### Looking Ahead

Existing software supply chains remain extremely complex, particularly as vendors adopt different business models for individual product lines. It is becoming increasingly difficult for companies to strike a balance between managing product development and distribution costs while ensuring a steady revenue stream from product licenses. We propose the following recommendations to help software product companies quickly optimize their software supply chains and generate additional revenues:

---

**Figure 4**

<table>
<thead>
<tr>
<th>Typical Entitlement Gaps</th>
<th>Embedded Software</th>
<th>Subscription Software</th>
<th>SaaS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Association of software with hardware</td>
<td></td>
<td></td>
<td>NA</td>
</tr>
<tr>
<td>Integration with service contracts</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Geo/site validation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Upgrades and renewals management</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend:
- **Low**
- **Moderate**
- **High**
1. **Embrace better code management**: Managing traceability and full visibility of different sources of code segments can help companies prevent the risk of impending legal issues or a negative impact on product branding.

2. **Improve the renewals processes**: By focusing on maximum renewals for existing sets of product licenses and investing in efficient entitlement management and provisioning, companies can minimize revenue leakage and identify renewal opportunities. This will also present opportunities for upselling, cross-selling and cycle-selling (e.g., month-end, quarter-end, etc.).

3. **Enhance license entitlement management**: Within entitlement management, companies should have adequate governance processes that address grace periods for licenses and support contracts, track entitlement support and manage license cancellation/termination to further reduce revenue leakage.

4. **Strive for more efficient SKU management**: The different permutations and combinations of product, price, geography, features, etc. make it difficult for companies to manage product SKUs. Therefore, they should ensure that the SKUs are rationalized on the basis of a global core with regional flexibility. An efficient bill of materials management engine can help achieve this.

5. **Move toward a pure SaaS model**: Making software products available on-demand instead of through physical fulfillment can help simplify supply chain complexities. Although the company will have to invest significantly in license entitlements and subscription management systems, many other complexities will be dramatically reduced, such as fewer product versions to support, more choices for deployment infrastructure and more direct customer touchpoints.

---

**About the Authors**

Charlie Goldenberg is Vice President of Cognizant Business Consulting (CBC), with responsibility for management consulting services delivered to the technology industry. He has more than 30 years of management consulting experience and was previously a partner with KPMG Consulting and Deloitte Consulting. At Deloitte, Charlie led the U.S. high-technology industry practice. He also was responsible for several of Deloitte's most important technology client relationships, including Microsoft, Sun Microsystems, Cisco and Intel. After leaving Deloitte, Charlie was a partner with Mercer Management Consulting, where he helped establish relationships with several major technology clients, including Symantec, Intel and HP. He started his consulting career with Booz & Co. Charlie can be reached at Charles.Goldenberg@cognizant.com.

Vivek Bhardwaj is a Manager within Cognizant’s Business Consulting Practice and has extensive product management experience with online companies in e-commerce, online products and supply chain management domains. He can be reached at Vivek.Bhardwaj@cognizant.com.

Chaitanya Pokkunuri is a Senior Consultant with the Cognizant Business Consulting Practice and is focused on helping technology companies with customer experience and customer support strategies. He can be reached at NVRKChaitanya.Pokkunuri@cognizant.com.

**About Cognizant Business Consulting**

With over 3,400 consultants worldwide, Cognizant Business Consulting (CBC) offers high-value consulting services that improve business performance and operational productivity, lower operational expenses, and enhance overall performance. Clients draw upon our deep industry expertise, program and change management capabilities, and analytical objectivity to help improve business productivity, drive technology-enabled business transformation, and increase shareholder value. To learn more, please visit [http://www.cognizant.com/business-consulting](http://www.cognizant.com/business-consulting) or e-mail us at inquiry@cognizant.com.
About Cognizant

Cognizant (NASDAQ: CTSH) is a leading provider of information technology, consulting, and business process outsourcing services, dedicated to helping the world’s leading companies build stronger businesses. Headquartered in Teaneck, New Jersey (U.S.), Cognizant combines a passion for client satisfaction, technology innovation, deep industry and business process expertise, and a global, collaborative workforce that embodies the future of work. With over 50 delivery centers worldwide and approximately 166,400 employees as of September 30, 2013, Cognizant is a member of the NASDAQ-100, the S&P 500, the Forbes Global 2000, and the Fortune 500 and is ranked among the top performing and fastest growing companies in the world. Visit us online at www.cognizant.com or follow us on Twitter: Cognizant.