WHITE PAPER

Build or Buy?

Key Considerations in Implementing Software Usage Analytics

Table of Contents

What to Know Before Building Software Usage Analytics In-House

Development teams are challenged to build increasingly competitive products, but they must overcome difficult tradeoffs to do so. They must prioritize a mounting backlog of new feature requests, balance innovation with maintenance, manage growing technical debt, improve quality and usability, accelerate delivery, and reduce cost—all at the same time.

Given limited resources, developers must optimize their investments to reflect true customer needs—while also anticipating customers' emerging requirements. Suboptimal product roadmap decisions can be disastrous.

To manage these challenges, software companies need comprehensive, accurate, timely, and actionable information about how users engage with their software.

To get this information, they are increasingly turning to software usage analytics solutions that:

- Instrument software applications to capture meaningful data about how customers and prospects are using them
- Help analyze engagement data to gain actionable insights on how to prioritize product development to deliver better products, increase customer adoption and retention, and grow sales

As developers contemplate usage analytics, one early question they often face is: should we build or buy? This guide sheds light on this decision from the standpoint of the development and product management teams. It assesses issues ranging from time-to-market to the direct and indirect costs of development and ongoing operation. Finally, it considers an alternative to home-built systems.

Why Software Developers Need Software Usage Analytics

By offering fact-based insights, an effective software usage analytics solution can help developers make far better day-to-day and strategic decisions. It can help them quickly surface emerging problems that are interfering with adoption, such as UI/UX issues or platform-specific bugs. It can also help developers uncover new customer needs, assess how quickly trends are taking hold, and align product roadmaps with customer realities.

In recent years, software companies have pursued diverse strategies to answer questions like these. These have included web analytics tools such as Google Analytics and Microsoft App Insights (see "Limitations of Web Analytics" on page 12); qualitative sales/channel feedback; helpdesk support calls; surveys; and measurements of software updates or downloads.

Each of these options made sense at the time, but each has serious shortcomings. For instance:

- Helpdesk calls capture an unrepresentative slice of user experience, e.g. you tend to hear about
- showstopper problems, not successful workflows or features
- Survey respondents are invariably self-selected
- Sales/channel feedback tends to be anecdotal
- Download counts deliver crude top-level information, offering no insight into why customers did or didn't upgrade, or how they use the software

In all these cases, developers see just a piece of the picture. It's almost impossible to combine these disparate fragments into a coherent whole—so it's hard to rely on them for decision-making.

Key Questions Software Usage Analytics Can Answer

- Which feature sets do users actually work with?
- Which features are attracting growing usage, and deserve more investment?
- Which features should be grouped for maximum ease of use?
- Which legacy features can be safely abandoned?
- Where are users encountering performance or usability problems?
- Are new software reliability issues emerging—and if so, where and on what platforms?
- What trial user behaviors lead to purchase or abandonment?
- What runtime environments should we target?

Build vs. Buy: Main Planning Considerations

Software usage analytics systems are complex. As software companies know better than anyone, building, maintaining, and operating complex software is expensive and time-consuming. It requires specialized resources and technical expertise at each stage of the development lifecycle, from requirements through testing, deployment, and maintenance. It requires extensive collaboration within and often beyond the organization. Moreover, these investments may come at the expense of responding to feature requests or new opportunities.

Software usage analytics systems demand careful attention to data, networking, access, security, customer privacy, and legal compliance. They require substantial client, server, and communications infrastructure. Even if only instrumenting a few thousand installations, they must scale to capture and manage massive amounts of data—and provide reliable real-time reports and visualizations for decision-making, no matter how much data must be processed.

While some elements of a home-grown software usage analytics system can be based on third-party libraries and frameworks, introducing these into your development environment can add long-term cost, complexity, and maintenance challenges. It might require you to retain skills unrelated to your core mission and weaken your focus on building your product.

Development teams that create their own custom in-house solutions must address all these issues. The following sections drill down into key aspects of an effective software usage analytics system, and the effort required to build rather than purchase it.

Upfront Planning

Like any complex software project, building a software usage analytics system requires extensive upfront planning by senior-level product managers, engineering leaders, IT leaders, and others throughout the organization. Table 1 describes some of the issues planners must address:

Table 1. Upfront Planning Issues.

Task	Discussion
Defining the metrics and metadata you need to collect	Significant collaboration is needed among company stakeholders and data users to ensure that you capture the right data to support actionable reports required by various team members (e.g., machine information, install and runtime statistics, granular and feature usage data) without gathering too much "noise" (data you'll never use).
Determining when and how to collect telemetry within the runtime process	Developers must instrument the application to collect metrics around user actions—events that actually matter. They must also determine which metrics to collect for every event.
Specifying telemetry transmission details	Development teams must design log formats, communication protocols, processes, synchronization schedules, and caching and handling for offline usage.
Specifying server and database requirements, including security	If developers plan to host their own servers and databases, they must plan for hardware infrastructure that can scale, software, backup, firewalls, anti- malware, and other security. If they intend to use cloud services, they must evaluate what type of servers and services will be required, predict usage costs, and plan for scalability and security.
Specifying requirements for dashboards, reporting, and visualization	As discussed in the Data Visualization and Reporting section below, decision-makers and technical professionals must work together to clarify: ▪ What questions need to be answered ▪ Who will have access to what information ▪ How the system will support ad hoc exploration and discovery
Defining integration requirements	Software usage analytics delivers the most value when it is integrated with third-party systems used in other areas of business operations. These can typically include: ▪ Product download and auto-update systems ▪ CRM and Licensing systems ▪ Marketing Automation ▪ Other business intelligence systems This can only be done if the system can export data in a format usable by these systems, or sync with them via well-defined APIs. In new systems developed from scratch, such interfaces may have to be built.

Client Instrumentation and Telemetry

Instrumenting clients non-disruptively and establishing reliable, secure telemetry between the client and server are non-trivial development tasks.

For example, to establish unique user profiles and track usage trends reliably for each installation, and link installation profiles to download sources or marketing campaigns, developers must automatically generate machine fingerprints and user installation IDs. So, too, most development organizations will want convenient and reliable opt-in/opt-out mechanisms. Development organizations that build from scratch must design, create, test, and integrate mechanisms to handle a wide variety of tasks. For these reasons, home-grown telemetry solutions often require months of development time, with extensive testing and quality assurance. Table 2 reviews some of the tasks involved: tasks that help explain why home-grown telemetry solutions often require months of development time.

Table 2. Client instrumentation and telemetry issues.

Task	Discussion
Building a secure and lightweight client- server communication protocol with embedded security and encryption of sensitive data	Client-server communication should be reliable and lightweight, so it does not affect user experience. For sensitive data collection where developers want to favor security over ubiquitous coverage, they may wish to require HTTPS client-server communication. In anonymous and less-sensitive industries, they may want to permit a fallback to HTTP, tracking installations behind firewalls and gateway filters that may block HTTPS.
Extending the client-server protocol to handle proxies, firewalls, web filtering gateways, dark networks, and other network configurations	As organizations seek to strengthen security, they are introducing more sophisticated means of network traffic control. Software telemetry systems must continually account for these environments in order to capture comprehensive data.
Building a method to generate unique customer IDs to track usage by user/ install even when tracking in anonymous mode	Developers need to provide support for reinstall, upgrade, multiple installs per machine, and other events. They must also link events to individual customer IDs in order to build the sequenced user profile histories that are critical to understanding how individuals use software.
Building logic to aggregate, compress, and optimize sending of telemetry data	Developers must invest significant time and effort in minimizing the frequency and size of call-home traffic, to ensure that telemetry doesn't interfere with client or network performance.
Building logic to handle communication errors such as "network not available"	To prevent data from being lost to communication errors or when users go offline, developers must implement intelligent local caching to transparently and securely store data locally until it can be forwarded to a cloud server.
Extending support to new platforms	When developers introduce products on new platforms, they must often repeat many of the development tasks described above.

Data Collection and Management

A successful software usage analytics system must efficiently collect and process massive datasets, making them available for advanced reporting and analysis. Timing information collection and providing adequate data storage and management can be daunting.

Development organizations with less data-intensive products often possess basic data management skills. For example, their developers may well have experience designing relational database schemas using SQL queries. But these basic skills aren't enough to cope with usage analytics applications that quickly generate terabytes of data, even if they are only collecting from a few thousand installations. So, for example, simple relational databases such as MySQL might fail to perform when you attempt to generate scalable real-time reports—requiring the use of alternative database design methodologies and visualization frameworks specialized for handling big data. These may be out of scope for your software product's development team. Table 3 identifies key issues involved in server- side data collection and management.

Table 3. Data collection and management issues.

Task	Discussion
Acquiring database and server hardware and software licenses	If you manage data locally, you'll need to provide or purchase licenses for the relevant databases and underlying server operating systems, unless you subscribe to a Platform-as-a-Service (PaaS) solution.
Building a database schema	This involves transforming the telemetry event types and parameters you've defined into a working database schema that can be queried efficiently for reporting.
Providing flexibility in what can be tracked	Some solutions limit flexibility in choosing what to track. Tracking events you don't care about can increase both out-of-pocket costs and operational complexity, and make client-server communications heavier than required. However, providing granular capabilities may involve additional work.
Supporting remote changes to tracking "on the fly" without requiring client updates	Home-built systems are often static: they always track the same events. To change what you track, you must change the client code and distribute a new build.A better alternative is to provide remote on/off control over tracking specific elements, ideally via a dashboard that non-technical users can operate. Companies that build their own systems should plan on coding and maintaining this additional functionality.
Capturing application-specific data (e.g., user-specified configuration settings; application state when an event occurs; user feedback sent via in-app messaging	These valuable capabilities are difficult to build on your own, and are not provided through commonly-used web analytics systems such as Google Analytics or Microsoft App Insights.
Scaling data storage	As discussed elsewhere, software usage analytics databases grow rapidly. Additional local or cloud resources may need to be acquired and managed; software companies may need to change underlying platforms to accommodate requirements for growth and performance.
Administering the data collection service and database system on a day-to-day basis	Like any web service or large database system, custom-built software usage analytics databases require ongoing monitoring, maintenance, and updates.

Data Visualization and Reporting

Your goal is not merely to capture and display data: it is to get timely, accurate answers you can use. As you may know from experience, if you require non-technical colleagues to write SQL queries, manually join datasets, or build complex Excel PivotTables, either your system will fail or you'll be overburdened with demands for support. Therefore, reporting should be simple and intuitive for non-experts.

Since software usage analytics will be valuable to a wide spectrum of executives, managers, and developers, systems must incorporate role-based reporting that is easy to manage.

Reporting should also operate in real-time, so technical and business decision-makers can act immediately. For example, if a new bug is preventing a subset of users from successfully upgrading, you need to know right now, so you can fix it immediately, before it affects more of your user base and generates massive numbers of support calls. So, too, imagine that days before a major product release, an executive team discovers a potentially significant issue affecting a subset of its user base. To choose the best course of action, decision- makers need immediate, reliable information about how many people would be affected if the release moves forward.

With huge datasets generated by software usage analytics, reporting can be painfully slow unless the analytics engine as well as supporting infrastructure, database, and queries have all been built for the purpose and carefully optimized.

Interactive visualization is often the most effective way to communicate software usage analytics insights. Many software organizations, recognizing the value of a powerful and flexible visualization dashboard, envision building an in-house solution based on a generic visualization framework.

Development organizations often underestimate the staff resources required to manage reporting and visualization, especially if these are not organizational core competencies. Moreover, third-party visualization tools typically assume your backend has enough resources to handle the query load.

Most important, a visualization framework merely lets you convert raw data into attractive charts. By itself, it doesn't add intelligence or context, or answer the questions that matter. For that, development teams need actionable interactive reports designed to answer specific questions. To build such reports from scratch, developers and data analysts must collaborate to understand context, choose metrics, correlate raw data, and build reports that help others make sense of it.

Building such actionable reports will require time, and knowledge of the framework. It will also require deep knowledge of what each potential user needs to know, both now and in the future, as they start asking more advanced BI questions. Some of the key issues associated with implementing visualization are listed in Table 4.

Table 4. Data visualization and reporting issues.

Task	Discussion
Implementing a modern dashboard that is customizable for each user persona needing access to data	Unless every person in your organization gets the data they need efficiently without a steep learning curve, they will probably end up avoiding your analytics framework, and fail to benefit from it.
Purchasing and implementing a visualization framework, and mastering its API	Developers must choose a visualization framework, master its API, and maintain those skills as long as they use it. Additional licensing, server, and administration costs are likely associated with the preferred toolkit. This toolkit must also be kept updated to support new browsers and JavaScript frameworks.
Designing high-performance backend queries	Development organizations sometimes underestimate the effort required to optimize backend queries so they are fast enough to scale while supporting live visualization in a timely manner that is practical for day-to-day use.
Implementing reports, including drill-down functionality	Drilldowns and related features help business users explore data in depth, and answer new questions that aren't already built into their reports, without requiring developers to create new reports.
Implementing data aggregation logic capable of identifying trends and patterns	This logic is required to make correlations and counts for dashboards and reports.
Implement data export capabilities	This is essential to integrate usage intelligence data with third-party tools to apply context and provide global business value.

Data Protection and Privacy Considerations

Development organizations know data protection and privacy are now of critical importance: failures can expose the business to extensive legal and reputational damage. Experience with usage analytics makes it easier to avoid trouble: experience with the technical aspects of building these systems, testing and deploying them, and operating them with real customers around the world. Table 5 outlines key issues associated with data security and privacy.

Table 5. Data protection and privacy issues.

Task	Discussion
Providing for data security and access control at all levels	Software companies that build their own usage analytics systems must protect customer and proprietary data at the client, in transit, at the server, and in use by company personnel. This often involves access management and encryption.
Offering appropriate notice and opt-in/opt-out options	Software companies that build their own usage analytics systems must develop appropriate opt-in/opt-out functionality in both the product UI and client tracking logic that can be updated as privacy rules change, or as you evolve data collection strategies.
Adapting systems for each region's rules	Developers must understand regional differences, such as the EU's stricter approach to consumer privacy. They may need to localize data storage, ensuring that customer data never leaves a specific region.
Complying with changes in privacy rules on a timely basis (e.g., the requirement to demonstrate compliance with the European Union's new General Data Protection Regulation (GDPR) by May 2018).	Development teams that build their own software usage analytics systems will need to track and respond to changes in privacy rules wherever they do business, or risk costly penalties.

Deploying a Robust, Well-performing Solution Environment

Beyond data, several more issues drive time-to-value, reliability, manageability, and overall ROI. Development organizations that build their own systems must address the issues shown in Table 6 below.

Table 6: Other key issues.

Task	Discussion
Non-disruptive API-based integration of usage analytics into company processes and third-party systems	With a software usage analytics API, it's easier to leverage useful software usage analytics data in diverse systems, including sales, marketing, business intelligence, and licensing frameworks—gaining a more holistic view of how users work with your software and respond to your marketing. Again, however, if you're building a custom system, you also need to provide a syncing mechanism to join data intelligently and inject only the specific data that should be provided to these external solutions. This needs to be carefully planned in advance.
QA, stress testing, security, and other factors	Developers must comprehensively test all their subsystems for reliability, performance, and security, including: ▪ The telemetry system ▪ Client-side SDKs on various client platforms ▪ Server-side infrastructure (using stress testing frameworks and tools) ▪ Server-side application (including telemetry user interface, reporting, and visualizations)
Resilience features	Software usage analytics is business-critical. Developers must provide capabilities expected in most modern business-critical systems, such as: ▪ Secure backup and recovery ▪ Automatic failover ▪ Accommodate for usage spikes and uneven traffic patterns (such as those associated with new version introductions) ▪ 24x7 server and service monitoring
Support for in-app messaging	Often, the best way to address the issues surfaced by software usage analytics is to communicate with your users directly within the application itself, via in-app messaging. If you don't purchase a software usage analytics solution that already includes this functionality (such as Usage Intelligence), you must build it yourself or purchase it separately and combine it with your analytics system.

Limitations of Web Analytics

Some software companies have attempted to apply web analytics frameworks to track desktop application usage, paying a monthly fee to utilize services such as Google Analytics or Microsoft App Insights. These software companies envision that a strategy built around web analytics will eliminate costs associated with hardware/software acquisition and continuous uptime monitoring. Unfortunately, leading web analytics platforms were not designed to address desktop software usage analytics. This has often extended development times, and resulted in failures to capture essential metrics or complete information about users' journey with a software application.

For example, Google Analytics focuses on pageviews, and can't efficiently track desktop applications. To track what happens after a product is downloaded, developers must build all client-side tracking code and manually map each captured action to a pageview or event with a unique URL. Each time they want to track an event, they must make a web request. The software must constantly maintain a live internet connection, generating internet traffic that can degrade user experience or raise suspicions of spyware infection. To support intermittent internet connectivity (such as usage on laptops) the developer must also build the client side logic to cache usage data while users are disconnected.

Moreover, since Google Analytics can't maintain user profiles, or generate reports spanning multiple sessions, developers can't track an individual's changing behavior over time without complex workarounds.

Microsoft App Insights was built to help organizations understand the performance of web-based applications running on the Microsoft Azure cloud platform, and identify root causes of cloud application problems. However, since it lacks key elements required in distributed Windows, macOS, or Linux applications, significant engineering resources are needed to collect, visualize, and analyze usage data. As requirements grow more complex, development and maintenance costs can soar.

Usage Intelligence: The Better Alternative

As a professional software development organization with a proven track record, given enough time and resources, you probably can build a software usage analytics system if you really want to. However, as this paper shows, doing so successfully is no small task. In some respects, it may be as challenging as building your own products.

Significant time, money, and resources are associated with building home-grown usage analytics with the sophisticated telemetry, data management, reporting, visualization, security, and performance you need. Going it alone may require you to acquire and maintain skills you don't possess and aren't relevant to your core mission.

Fortunately, there's a much easier, faster, and lower-cost way to get the usage analytics you need: Usage Intelligence. Working with Usage Intelligence enables you to rapidly bring a solution into production—often, with your next point release. For example, Usage Intelligence gives you:

- Comprehensive client, server, data, communications, and security infrastructure—all integrated, fully-tested, quality-assured, and optimized
- A convenient API that lets you instrument your software and start collecting data in as little as 30 minutes, with just 10 lines of code
- Native support for Windows, Linux, and macOS, with native APIs for C/C++, .NET, Objective-C and Java, so you can instrument nearly any desktop application
- A client solution providing for various scenarios, with inbuilt caching, offline tracking, and reliable machine identification to maximize tracking capabilities on the client side
- Comprehensive reporting right out of the box, reflecting the key metrics and KPIs that software companies have found most valuable, all optimized to run responsively in real time A modern interactive dashboard with role-based security, enabling less technical users to dynamically visualize and explore data out-of-the-box, and answer business questions without developer or tech support
- Straightforward integration of data and visualizations into other processes and third-party systems via a proven SDK and published APIs
- Built-in support for capturing architecture, platforms, and geolocation information to help you understand usage patterns by user profile or region
- Built-in in-app messaging, so you can communicate individually with users, solve emerging problems, and point out new features and opportunities
- Virtually unlimited scalability
- Access to Revenera's extensive expertise about all technical, business, privacy, and legal aspects of software usage analytics, reflecting our years of experience working with hundreds of products and companies worldwide

NEXT STEPS

Discover how Revenera can deliver the user insights you need to build your next great product.

LEARN MORE >

Revenera provides the enabling technology to take products to market fast, unlock the value of your IP and accelerate revenue growth—from the edge to the cloud. **www.revenera.com**

revenera

1.800.809.5659 | +44.870.871.1111 | revenera.com