Application Analytics for effective beta release management

In order to thrive, modern software development practices must keep pace with accelerating DevOps capabilities and increasingly diverse user platforms and devices. Effective and efficient beta/preview release management is a critical step on this journey.

Benefits of effective beta release management

1. **Gauge customer experience and satisfaction**: feature discovery, user behaviors, and usage patterns of new and legacy features are measured and validated improving adoption, satisfaction, and impact in production.

Aggregate analytics

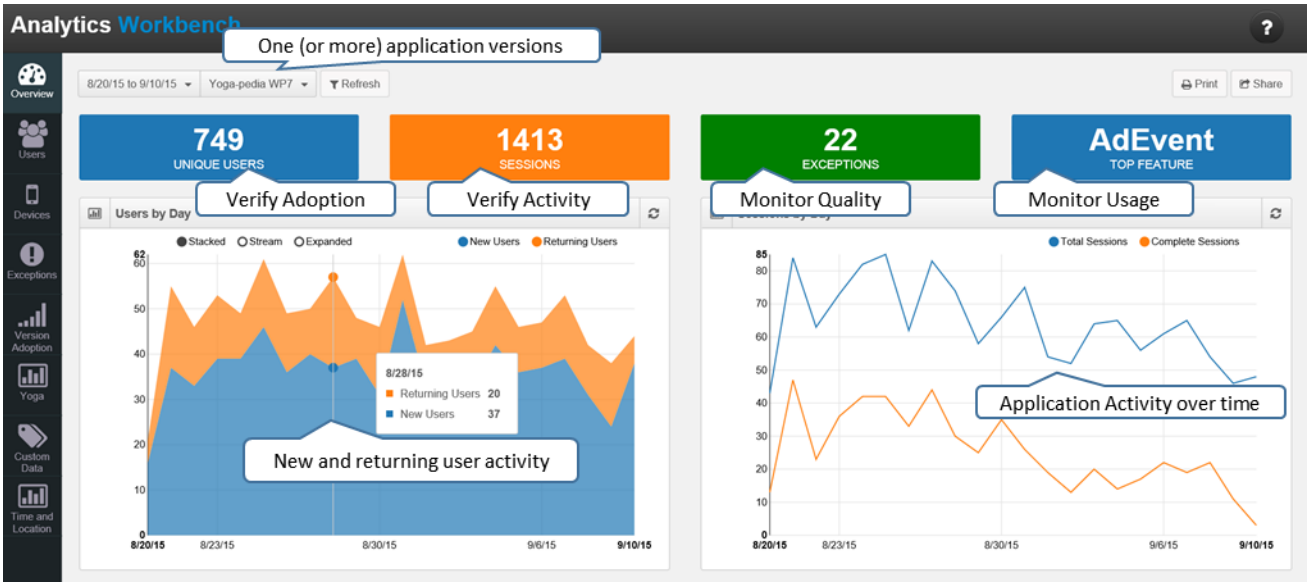

Figure 1: *PreEmptive Analytics Workbench Overview page*

Figure 1 shows an aggregate view of user adoption, application activity, quality metrics, and feature usage – cut across a single version of an application within a discrete timeframe, e.g. a beta release of an application.

Figure 1a: *PreEmptive Analytics Workbench Overview page detail – aggregate feature usage*

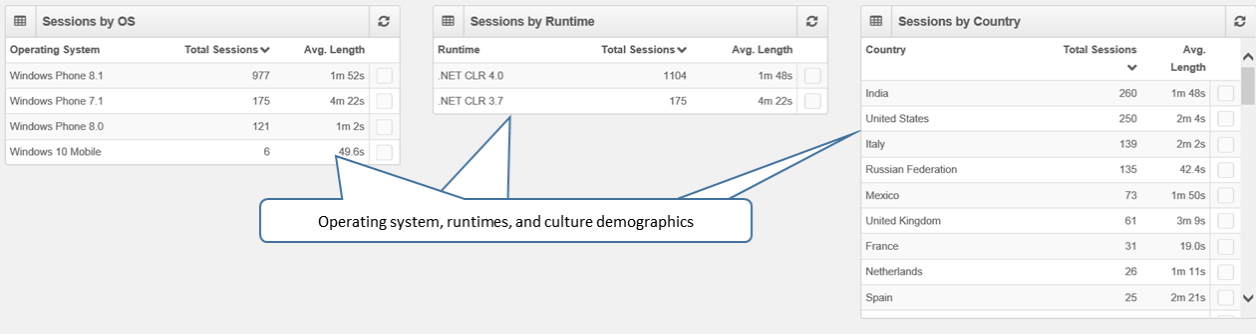

Figure 1b: *PreEmptive Analytics Workbench Overview page detail – aggregate operating system, runtime, and geographical/cultural distribution across application*

Figures 1a and 1b provide further aggregate insights into where and how an application is being used.

Individual user analytics

The PreEmptive Analytics Workbench can associate a given data element (e.g. User Id) with every other data element inside a Workbench (e.g. Feature or Exception) so that the Workbench can pivot and filter based on that first element (e.g. "show all features that were used by User Id xxx"). This capability is called a “Pattern” and the Beta Release Management configuration of the Workbench shown here includes a User Id “Pattern” where a User Id maps to either a user’s device or a user’s credentials across multiple devices.

This extension allows a Workbench user to look at quality, usage, and behaviors down to the individual user – *an essential diagnostic and analytics capability*.

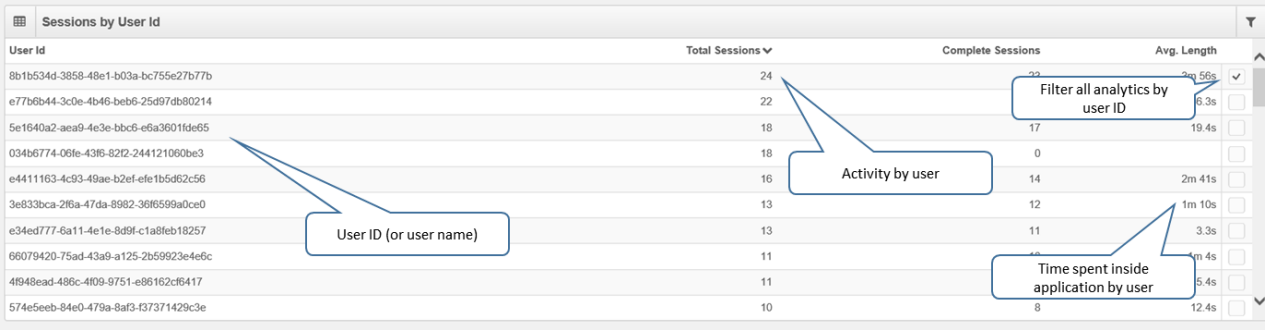

Figure 2: *PreEmptive Analytics Workbench Users page detail*

Figure 2 shows session counts and average session lengths by User Id. In Figure 2 above, the most active User Id (the top row) has been selected (see upper right hand corner where the “box” has been checked).

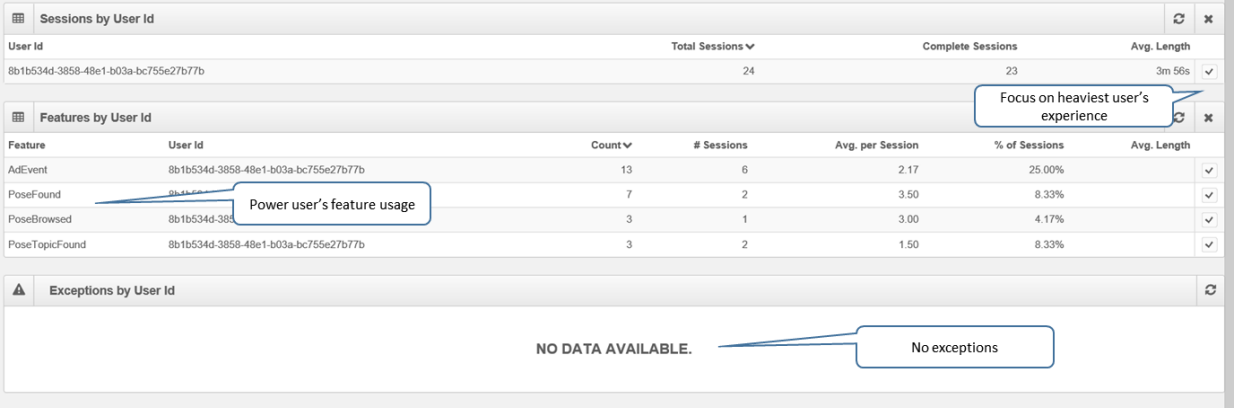

Figure 2a: *PreEmptive Analytics Workbench Users page*

Figure 2a shows the feature usage the one "Power User" selected in Figure 2 above. It also shows that this particular user did not experience any application exceptions at any time across the 24 sessions recorded.

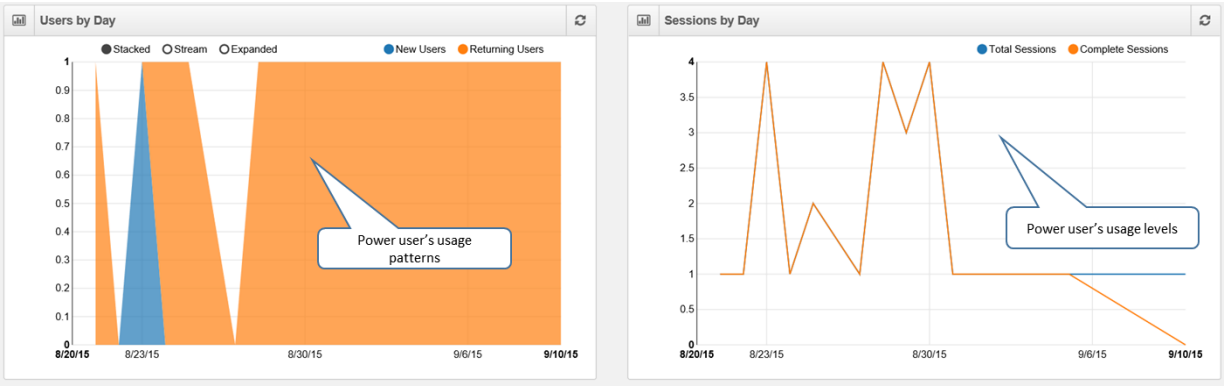

Figure 2b: *PreEmptive Analytics Workbench Overview page after User-Id filter has been applied*

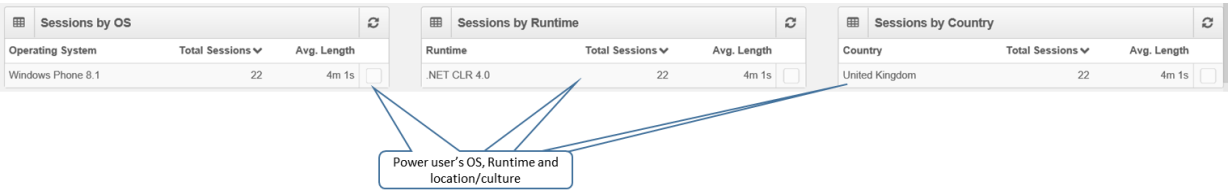

Figure 2c: *PreEmptive Analytics Workbench Overview page detail after User-Id filter has been applied*

Figures 2b and 2c show how the User Id pattern automatically applies across dashboards allowing the Workbench user to see session, activity, and even platform, runtime, and geolocation of the individual user selected in the Users page (see Figure 2 above).

Benefits of effective beta release management

2. **Measure quality and experience against diverse runtime environments**: real-world permutations of mobile, thick, and web surfaces across networks and domains cannot be reliably simulated and often produce unexpected results.

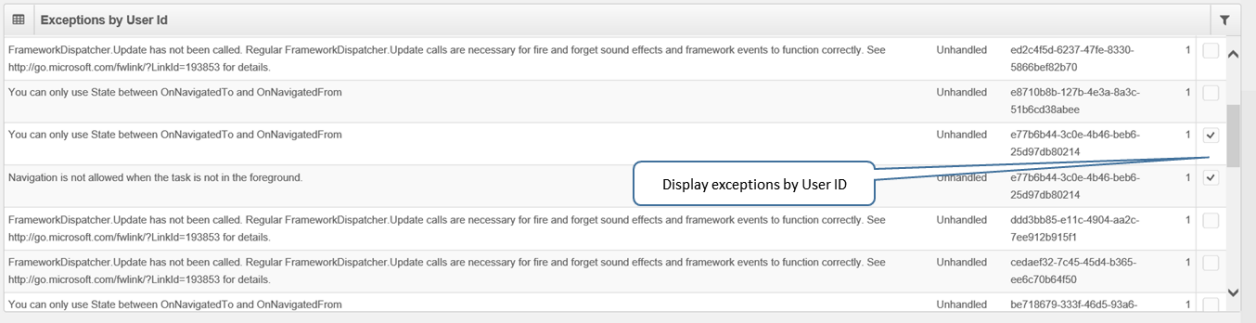

Figure 3: *PreEmptive Analytics Workbench Users page detail*

Figure 3 shows exceptions by User ID, the type of exception (unhandled, thrown, or caught), the specific exception and a count. Two exceptions associated with a specific User Id have been selected to filter all Workbench dashboards to focus on the user who experienced these specific exceptions.

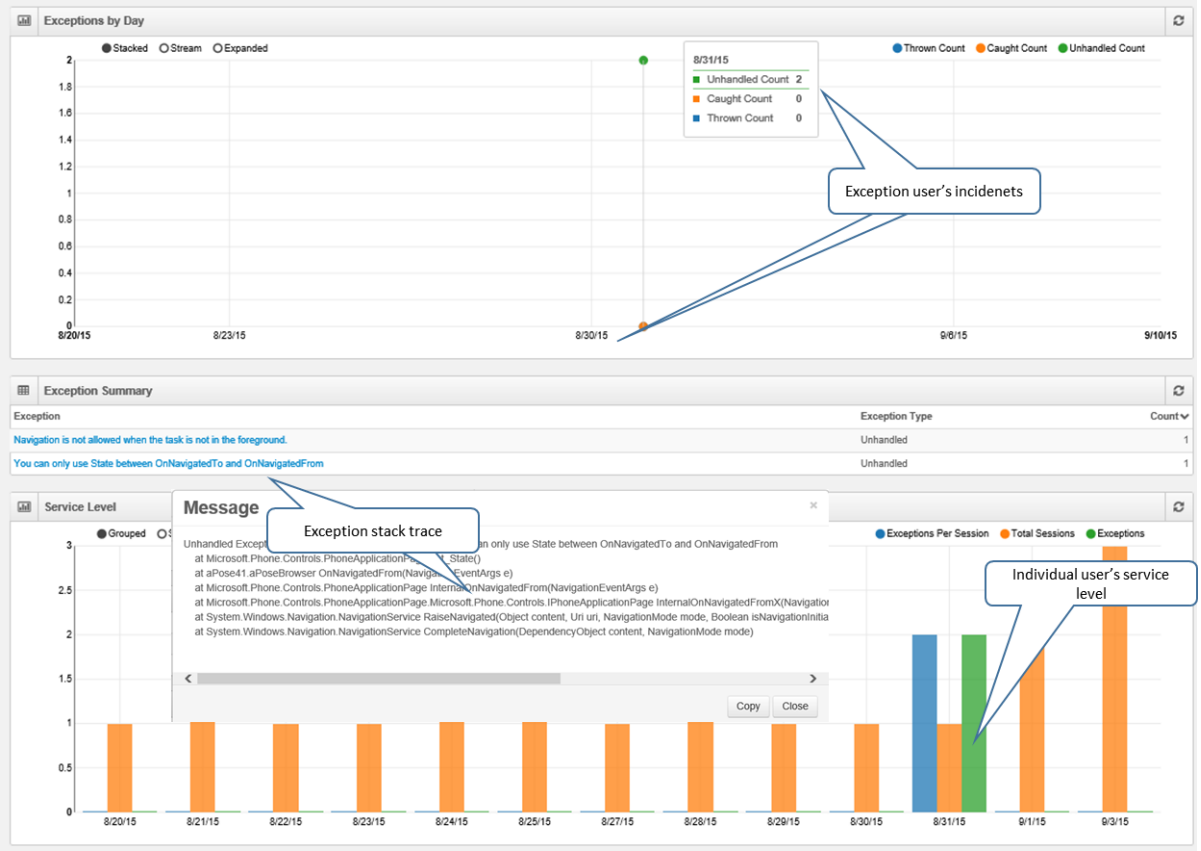

Figure 3a: *PreEmptive Analytics Workbench Exceptions page*

Figure 3a shows the timing, distribution, the stack trace, and the ratio of sessions to exceptions (an application owner's own definition of service level).

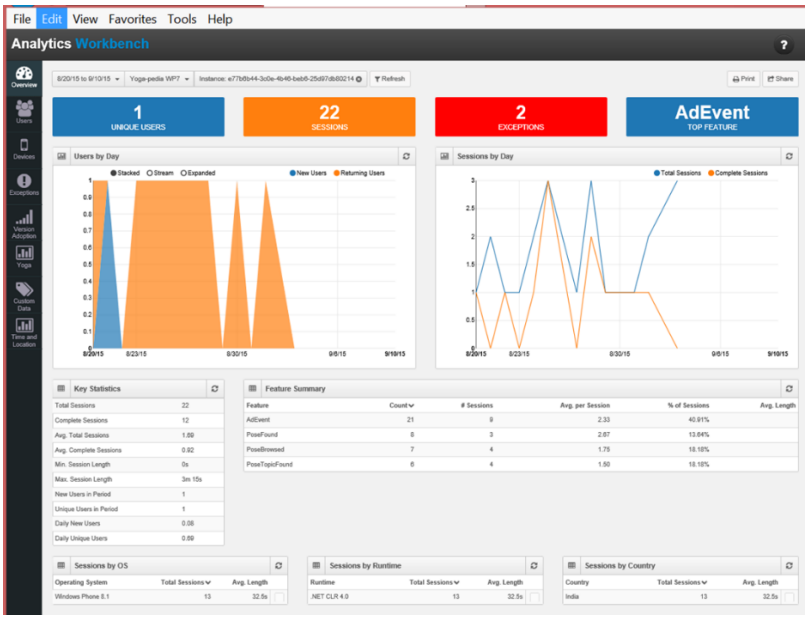

Figure 3b: *PreEmptive Analytics Workbench Overview page*

Figure 3b shows both the individual user pattern (features used, frequency, location, etc.) of the user who experienced the exceptions under investigation and the user's device model – factors that may contribute to the user's poor experience.

Benefits of effective beta release management

3. **Simplify issue replication and improve regression testing**: capture and ultimately reflect runtime and technology profiles of actual users into testing and release plans.

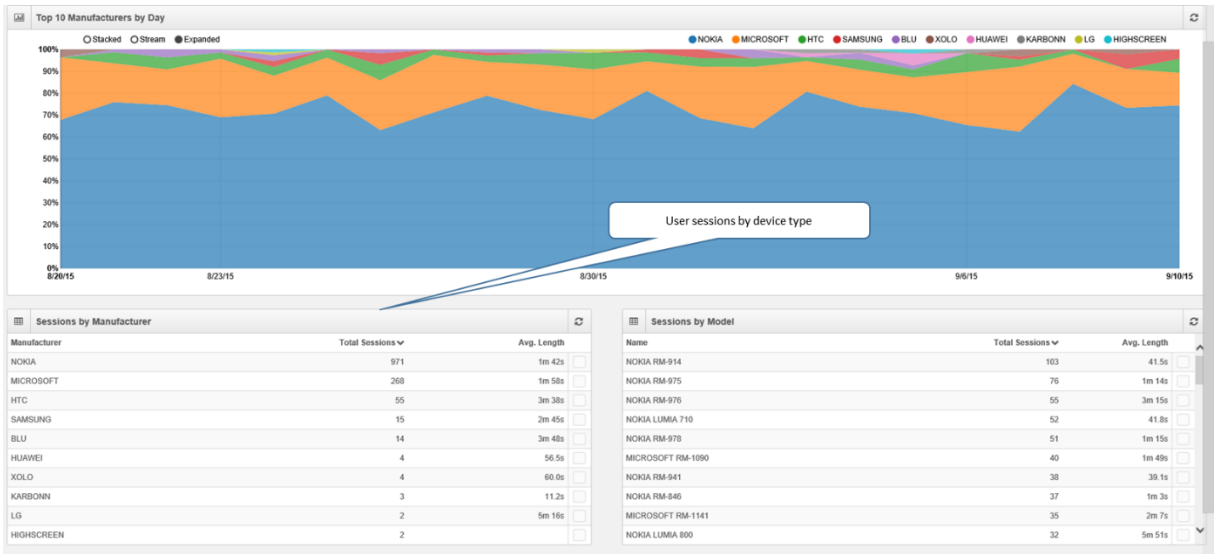

Figure 4: *PreEmptive Analytics Workbench Devices page*

Figure 4 shows the distribution of usage across all device manufacturers and devices. **Note: hardware stack of PC-based users and servers can also be retrieved – this is NOT a mobile-specific solution**.

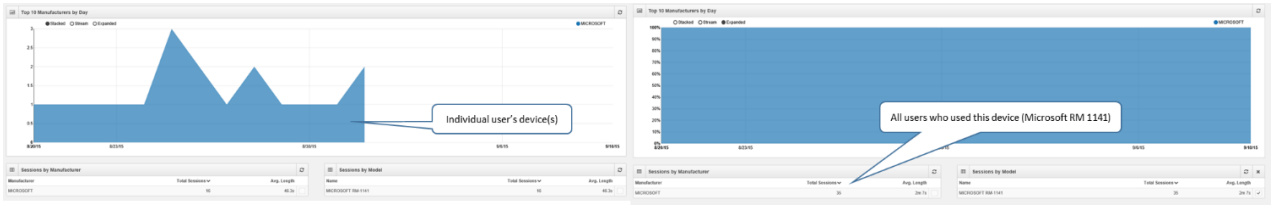

Figure 4a: *PreEmptive Analytics Workbench Devices pages side-by-side comparison*

Figure 4a shows the device model of the individual user whose exceptions were investigated in Figures 3 through 3b and the total number of sessions across all users who used the same model. **NOTE**: model is another analytics "Pattern" and can be used to filter data across all dashboards.

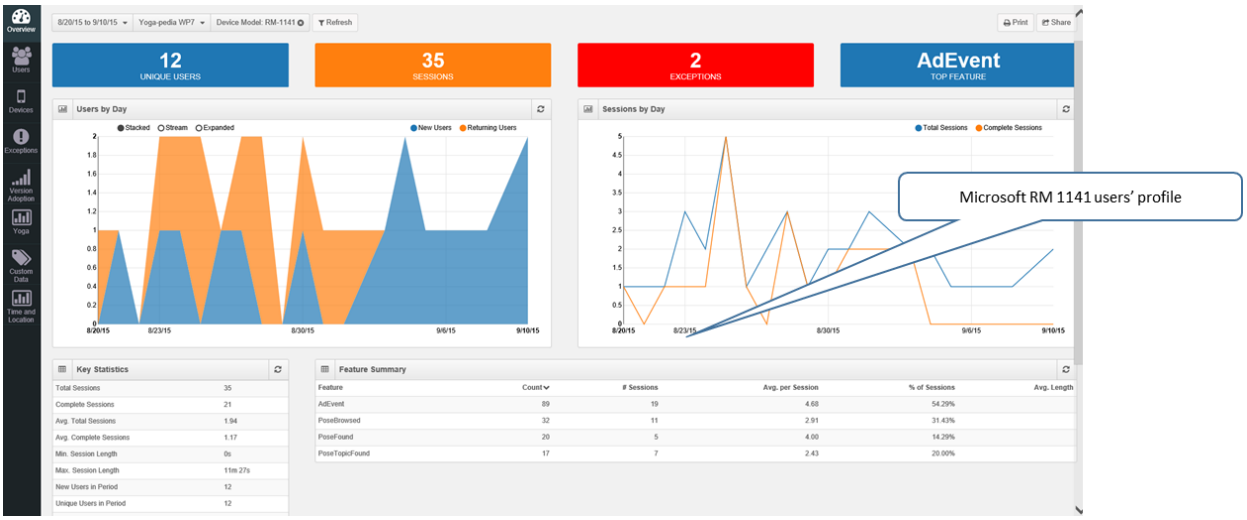

Figure 4b: *PreEmptive Analytics Workbench Overview page of all users using the same device model*

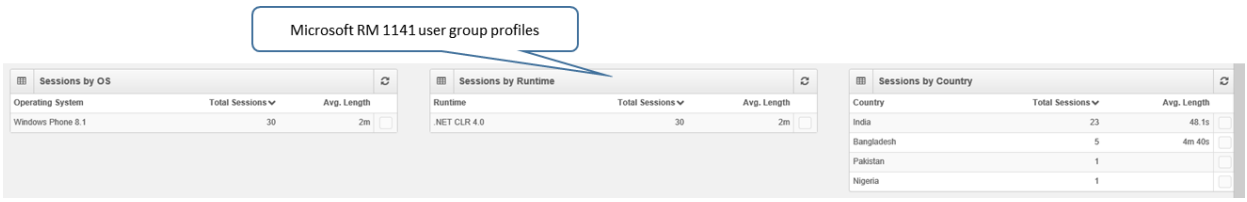

Figure 4c: *PreEmptive Analytics Workbench Overview page detail of all users using the same model*

Figures 4b and 4c show that there were a total of 12 users who used the application on the same model type, but only one user experienced a quality issue. It is likely that OS, device type, location, culture, etc. are contributing to the root cause of this issue.

Benefits of effective beta release management

4. **Assess infrastructure scale, performance, and stability**: measure assumptions against actual usage patterns across tiers.

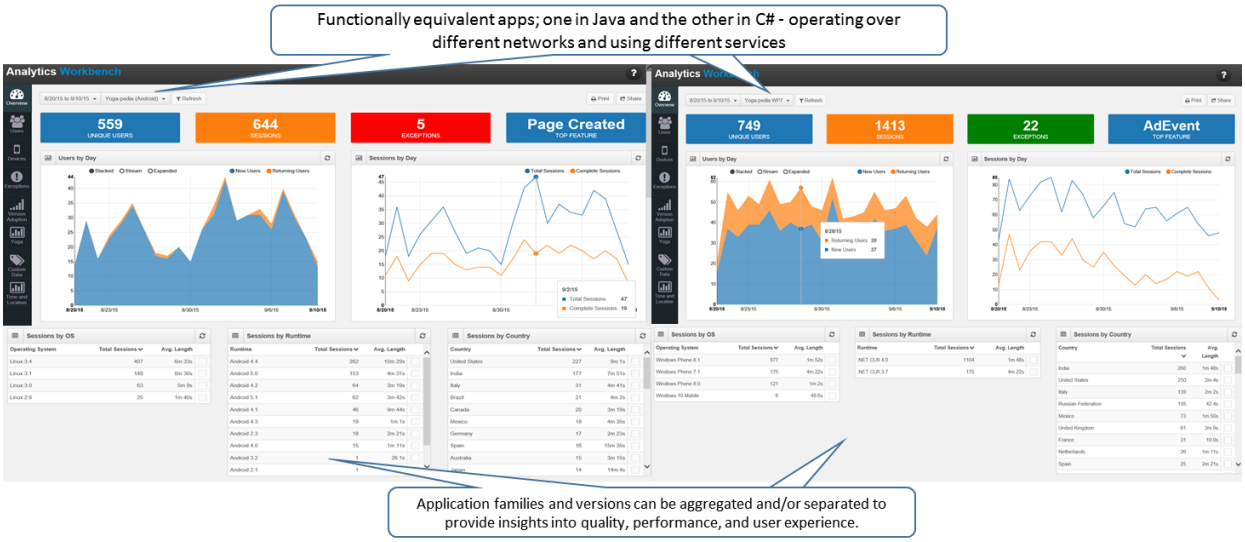

Figure 5: *PreEmptive Analytics Workbench Overview pages side-by-side*

Figure 5 shows C# and Java applications that are functionally equivalent but have been implemented in different languages, run on different hardware, and rely upon different software services. The PreEmptive Analytics Workbench simplifies side-by-side assessments on how related and/or alternative components compare, interoperate, and ultimately impact user experience and development ROI.

What makes a "better beta" and how do application analytics help?

1. **The larger the beta user community, the better (more valid) the feedback.** However, more users also mean more time spent coordinating, supporting, and managing those users.

Application Analytics removes the requirement to survey and record what users do with your software; now you only need to ask them about their impressions and overall satisfaction.

Application analytics:

- Improves usage recording accuracy
- Reduces effort/commitment required of both users and beta support staff
- Increases user participation by lowering their required time commitment

2. **The earlier you can collect feedback from users, the greater its impact.** Yet, care must be taken to ensure that the effort to build and manage beta releases sooner and more often does not delay/impede development schedule commitments.

Injection of application analytics instrumentation post-compile frees development from having to instrument their application reducing the cost and effort required for each release. Further, the remaining effort is managed entirely outside of core application development.

Application analytics instrumentation injection:

- Delivers quality feedback earlier in the development cycle
- Reduces the overall cost of preview/beta releases
- Reduces development burden
- Accommodates product owner/manager feedback requirements without impacting development

3. **Running multiple, iterative beta cycles gives developers the chance to address problems as they're found and verify that outstanding issues have been addressed.**

Application instrumentation injection patterns can be adjusted to accommodate evolving analytics requirements in-step with evolving application iterations.

Application analytics instrumentation injection:
- Accelerates release velocity
- Validates fixes and design changes are effective before they're in production
- Supports role-specific dashboards and KPIs aligned with each beta iteration

4. **International distribution of beta releases validate assumptions across diverse runtime and cultural contexts.**

Dynamic and configurable opt-in and telemetry distribution options ensure that application analytics telemetry is collected in compliance with governing opt-in, privacy, and compliance obligations – even across diverse jurisdictions and regulatory frameworks.

Dynamic and configurable opt-in and telemetry distribution options:
- Enforce company and jurisdictionally dependent opt-in policy(ies)
- Support the distribution and management of telemetry according to in-house privacy and governance policies

5. **Piracy, Intellectual Property (IP) theft, counterfeiting, and other material application risks must be managed** to protect development investments and reduce other operational and reputational risks.

PreEmptive Application Analytics also includes obfuscation to prevent reverse engineering, **tamper defense** to prevent counterfeiting and malware injection, and **auto-expiry policy enforcement** to enforce application inventory control without coding or license management overhead. Further, **the injection of instrumentation can be omitted** during production fully removing all beta-centered instrumentation.

PreEmptive Analytics application hardening capabilities:
- Secure beta releases against piracy, tampering, and other malicious and unauthorized use cases without increasing development burden.
- Post compile-injection can be "switched-off" for production builds removing all instrumentation (and any privacy or governance concerns that these may convey).