Usability for high code software to help improve contributions

Aparna Sundar
Senior Researcher
*OpenSearch Project*
Do hi-code users need usability?
Agenda

• User Experience based on perception
• Thinking through a usability study
• The Who
• Goals of the study
• Study set up
• Task demo
• Findings
• Next steps
Agenda

• User Experience based on perception
So from a UX perspective, what is UX?

The user experience is how a user interacts with and experiences a product, system or service. It includes a person's perceptions of utility, ease of use, and efficiency.
Popular perception: Open source software is less usable

- Open source developer-user who both uses the software and contributes to its development
- User-centered design movement attempts to bridge the gap between programmers and users through specific techniques (usability engineering, participatory design, ethnography etc.)
Agenda

• User Experience based on perception
• Thinking through a usability study
Usability studies

Observational methodology to uncover problems and opportunities in designs

Potential to address other use cases
How to Think Through the Study

- The user is important
- Think through user flow

User Flow Design

1. User goal
   Goal or story level

2. Task flow
   Action level

3. Wireflow
   Component level

4. User Flow
   Interaction level
Agenda

• User Experience based on perception
• Thinking through a usability study
• The Who
Four User Type by Role

Validated in 2023 Q1 OpenSearch User Survey

INFRA      ADMIN      PRODUCER      CONSUMER
Baseline for Admins

In this year’s OpenSearch Project Q1 community survey, 48.8% of community members self-identified as being in an administrative role (Admin) in OpenSearch3.

The Admin, is responsible for asset and user management in OpenSearch, and has a critical role in determining the quality of experience and usability of users downstream, such as Producers and Consumers.

In the study, we presented 20 participants with seven tasks from the Admin panel workflow.

The seven tasks cover the most common data management functionalities, such as managing and creating indexes and index lifecycles, in OpenSearch.
Agenda

• User Experience based on perception
• Thinking through a usability study
• The Who
• Goals of the study
Goals of the Study

• Baseline for Admins for future improvements to UX
• In this year’s OpenSearch Project Q1 community survey, 48.8% of community members self-identified as being in an administrative role (Admin) in OpenSearch3.
• The Admin, is responsible for asset and user management in OpenSearch, and has a critical role in determining the quality of experience and usability of users downstream, such as Producers and Consumers.
Agenda

• User Experience based on perception
• Thinking through a usability study
• The Who
• Goals of the study
• Study set up
Participants

- 18 hi-code admin users
- The seven tasks cover the most common data management functionalities, such as managing and creating indexes and index lifecycles, in OpenSearch
Agenda

- User Experience based on perception
- Thinking through a usability study
- The Who
- Goals of the study
- Study set up
- Task demo
Welcome to OpenSearch
Dashboards

Start by adding your data
Add data to your cluster from any source, then analyze and visualize it in real-time. Use our solutions to add search anywhere, observe your ecosystem, and protect against security threats.

Add data  Explore on my own
Agenda

• User Experience based on perception
• Thinking through a usability study
• The Who
• Goals of the study
• Study set up
• Task demo
• Findings
<table>
<thead>
<tr>
<th>Task #</th>
<th>Task Description</th>
<th>Instructions</th>
<th>Success Path</th>
<th>#Succeeded</th>
<th>#Failed</th>
<th>#Succeed with Assistance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Create an index template.</td>
<td>Create an index template named “flight_data_template”. Include any indexes that start with the name flight_log. For index configuration, include the following 2 configurations: 1) Settings: Set primary shards to 2. Use default for everything else. 2) Mappings: Two new fields: “timestamp” and “flight_id”.</td>
<td>1. Index management &gt; Templates &gt; Create Template 2. Fill in name &gt; Specify pattern/Wildcard &gt; specify alias &gt; Index Mapping</td>
<td>17</td>
<td>2</td>
<td>Hint: Index pattern is flight_log*</td>
</tr>
<tr>
<td>2</td>
<td>Create an index with the settings that inherits from the template you just created.</td>
<td>Create an index with a name that matches the wildcard from the template you have defined.</td>
<td>1. Index management &gt; Indices &gt; Create Index 2. Fill in a name matching Task 1’s wildcard (User gets feedback when matches wildcard)</td>
<td>13</td>
<td>3</td>
<td>Hint: If user is stuck on naming, direct them to name the index flight_log_jan23 If user failed to create an index template in Task 1, tell them to create flight_log_jan23 with settings: 2 primary shards; default everything else and mappings: 2 new fields: timestamp and flight_id</td>
</tr>
<tr>
<td>3</td>
<td>Find the information of that index such as alias, settings, mappings</td>
<td>Look for index settings, mapping, and aliases.</td>
<td>1. Click appropriate index, scroll to bottom</td>
<td>20</td>
<td>0</td>
<td>No hint</td>
</tr>
<tr>
<td>4</td>
<td>Delete any flight log indexes once they are older than 21 days. To automate this, create a state management policy using the visual editor.</td>
<td>Create a policy named “reduce_indexes”. It will include any indexes matching the index pattern flight_log*. Configure the following 2 states: 1) Check if current indexes are older than 21 days, transition the indexes into the delete state. 2) Perform delete action.</td>
<td>1. Index management &gt; state management policies &gt; create policy &gt; visual editor 2. Specify policy name 3. Specify index pattern 4. Add state &gt; Name destination state ‘current’ or similar &gt; add transition &gt; set condition Minimum Index Age to 21d &gt; set destination state</td>
<td>4</td>
<td>13</td>
<td>If users are stuck/facing errors, remind them to create 2 states: one named ‘current’ to check the index age, and one named ‘delete’ to perform the delete action, If the indexes are older than 21 days, transition them to the delete state</td>
</tr>
<tr>
<td>5</td>
<td>Make major configuration changes to an existing index by reindexing an existing index to a new index.</td>
<td>Reindex from the index “flight_log_jan23”. Create a new index named “flight_log_jan23_updated”. The new index will need 4 primary shards.</td>
<td>1. Index management &gt; state management policies &gt; create policy &gt; visual editor 2. Specify policy name 3. Specify index pattern 4. Add state &gt; Name destination state ‘current’ or similar &gt; add transition &gt; set condition Minimum Index Age to 21d &gt; set destination state</td>
<td>16</td>
<td>4</td>
<td>No hint</td>
</tr>
<tr>
<td>6</td>
<td>Take a daily snapshot of any flight log indexes.</td>
<td>Create a policy named “daily_snapshots”. It will include any indexes matching the index pattern flight_log*</td>
<td>17</td>
<td>3</td>
<td>No hint</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Restore data from an existing snapshot.</td>
<td>Restore all indexes from a snapshot “flight_info_snapshot_12282022”</td>
<td>20</td>
<td>0</td>
<td>No hint</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Success Rate</td>
<td>Failed Reasons</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create an index template</td>
<td>17 / 20</td>
<td>User did not use wildcard (missed index pattern, filled in action and transition within one state)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create an index that inherits a template</td>
<td>13 / 20</td>
<td>Missed index pattern, transition is unintuitive, minimum index age error</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Review an index</td>
<td>20 / 20</td>
<td>Failed to drag action/transition to reorder</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create an index management policy</td>
<td>4 / 20</td>
<td>Missed index pattern, transition is unintuitive, minimum index age error</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reindex an existing index</td>
<td>16 / 20</td>
<td>Missed index pattern, transition is unintuitive, minimum index age error</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create a snapshot</td>
<td>20 / 20</td>
<td>Failed to drag action/transition to reorder</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Restore an existing snapshot</td>
<td>20 / 20</td>
<td>Failed to drag action/transition to reorder</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Task 1**: Create an index template
  - Failed: User did not use wildcard.
  - Success: Expected a dropdown of templates to choose from.

- **Task 2**: Create an index that inherits a template
  - Failed: Missed index pattern, filled in action and transition within one state, transition is unintuitive, minimum index age error.
  - Success: SWA (needed help with index name, expected ability to create index from template).

- **Task 3**: Review an index
  - Failed: Transition is unintuitive.
  - Success: Success (expected ability to create index from template).

- **Task 4**: Create an index management policy
  - Failed: Missed index pattern, filled in action and transition within one state, transition is unintuitive, minimum index age error.
  - Success: SWA (initially scrolled and missed the template confirmation).

- **Task 5**: Reindex an existing index
  - Failed: Missed index pattern, filled in action and transition within one state, transition is unintuitive, minimum index age error.
  - Success: SWA (initially skipped ISM, transition is unintuitive).

- **Task 6**: Create a snapshot
  - Failed: Missed index pattern, filled in action and transition within one state, transition is unintuitive, minimum index age error.
  - Success: SWA (initially skipped ISM, transition is unintuitive).

- **Task 7**: Restore an existing snapshot
  - Failed: Missed index pattern, filled in action and transition within one state, transition is unintuitive, minimum index age error.
  - Success: SWA (initially skipped ISM, transition is unintuitive).
Finding

Relatively high level of success with the interface

Of 140 total task attempts, there were 107 successes (~76%); 25 failures (~18%); and 8 successes with assistance (~6%)

Self service users on average succeeded on 90% of tasks in comparison to Managed service users who succeeded only on 78.6% of tasks
In the study we further explored how Admins undertake four tasks that do not have a user interface: cluster configuration, monitoring, data replication, and data ingestion management.

Participants fear data loss and downtime and do what they can to protect themselves and/or the clients that they manage from loss.

Participants also shared that they cared about OpenSearch itself does not go down, so they monitored OpenSearch and create a disaster recovery plan.

Ultimately, users need seamless integration with external software to conduct the tasks listed above and to back up their systems.
Agenda

• User Experience based on perception
• Thinking through a usability study
• The Who
• Goals of the study
• Study set up
• Task demo
• Findings
• Next steps
Opportunities for Improvement and Future Research

- Simplify complex tasks with interface modifications

- Find more unified solutions to prevent loss of data and/or downtime, focusing on migrating, security, and monitoring

- Implement interface guidance

- Consider implementing behaviors and placement of notifications that are consistent in order to improve the overall user experience
Thank you