The other end of Software development

Robert Hess

Quality Inquisitor

GoTo, Inc.

Aber wir haben uralte Software …

- Ganz ohne Tests, auf on-premise Servern, handgepflegt.
- In Programmiersprachen, die nicht besonders gut geeignet sind sicheren Code zu schreiben.

Nebenbemerkung: wir reden hier über SaaS Software.
Alles, was in Steuerungen, isolierten Kiosksystemen etc. verbaut ist, ist eine andere Sache.
Solange es nicht über das Internet oder anderswie erreichbar ist.

Die “Wahrheit” über Software re-writes

- Niemals
  - Software enthält Wissen
  - Re-write blockiert neue Features
  - Es geht wirklich oft schief.
- Manchmal doch
  - Parallel zum bestehenden Produkt,
  - Das alte Produkt weiterlaufen lassen.
  - Funktioniert nur in einem wachsenden Markt.
- Es wird einfach immer wieder versucht …

Die “Wahrheit” über das Alter

- Abgehangene Software – mit der Zeit werden alle offensichtlichen Bugs gefixt

→ klappt nur, wenn keine Änderungen

- Software an der entwickelt wird, erfordert Wartung – Umstrukturierung, Refactoring.
- Aufwand dafür steigt ~ mit der Größe der Software, abhängig von Codequalität und Architektur
- Um so mehr Code erzeugt wird, um so mehr Zeit wird prozentual für Wartungsarbeiten benötigt
- Im Extrem bleibt keine Zeit mehr, um neuen Code zu schreiben.
- Es gibt diverse Workarounds, aber keine wirkliche Lösung.

Wie kriegen wir raus ob unsere Software (immer noch) funktioniert?

- Testen
- Monitoren
- Nutzer fragen
- Schauen ob der Profit steigt

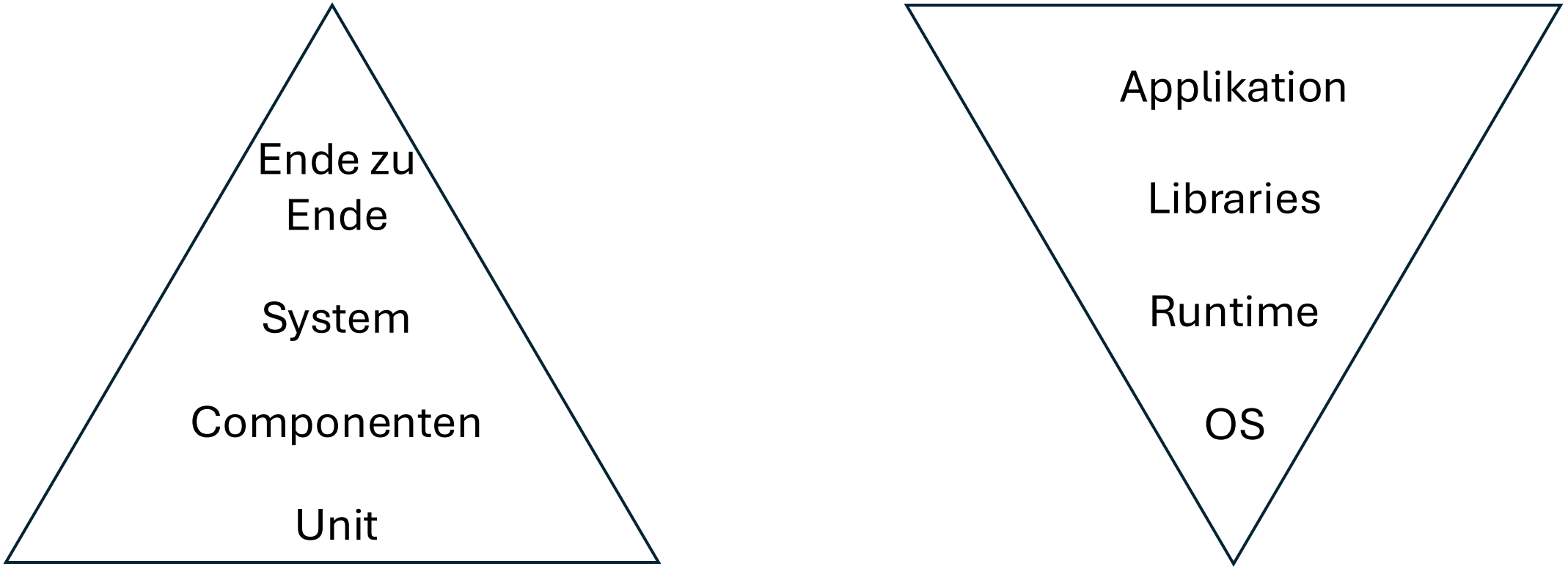
Was ist nun dieses Testen?

Testen

- Software, die die Benutzung der eigentlichen Software (der Software under Test, SUT) simuliert, und deren Verhalten überprüft.
- Menschen, die die Software under Test benutzen, und dabei Fehler hervorrufen (und davon berichten, ggf. automatisiert).

Sieht ähnlich aus, ist es aber nicht

Testpyramide ← → Komplexitätspyramide

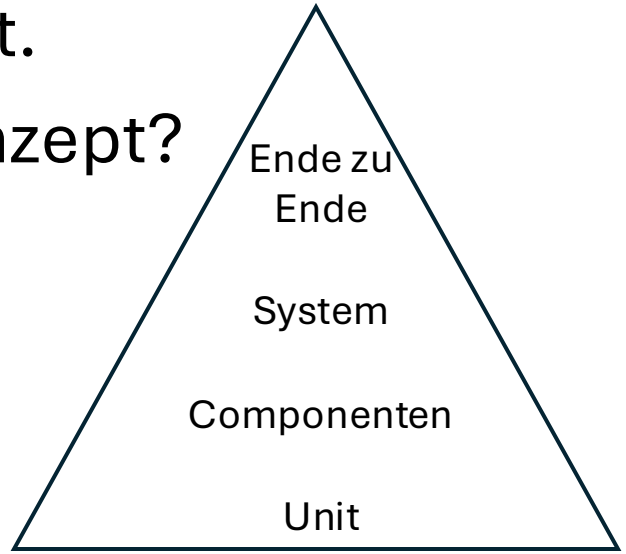


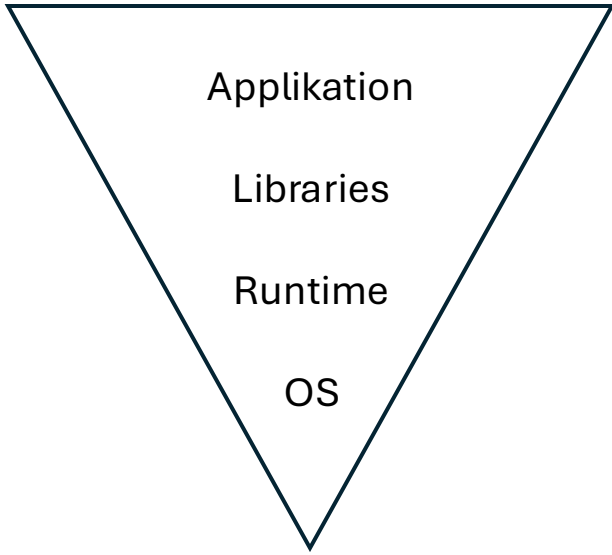
https://martinfowler.com/articles/practical-test-pyramid.html

Komplexität

- Komplexität wächst von unten nach oben
- Tests nach Testpyramide testen bevorzugt auf den unteren Ebenen, *weil es einfach und billig ist dort zu testen*. Dabei wird aber im wesentlichen der eigene Code getestet – nicht so sehr die Interaktion mit dem Rest der Welt
- → es stellt sich heraus, dass der Rest der Welt ein nicht zu vernachlässigender Faktor ist.

Wie kriegt man das ins Testkonzept?



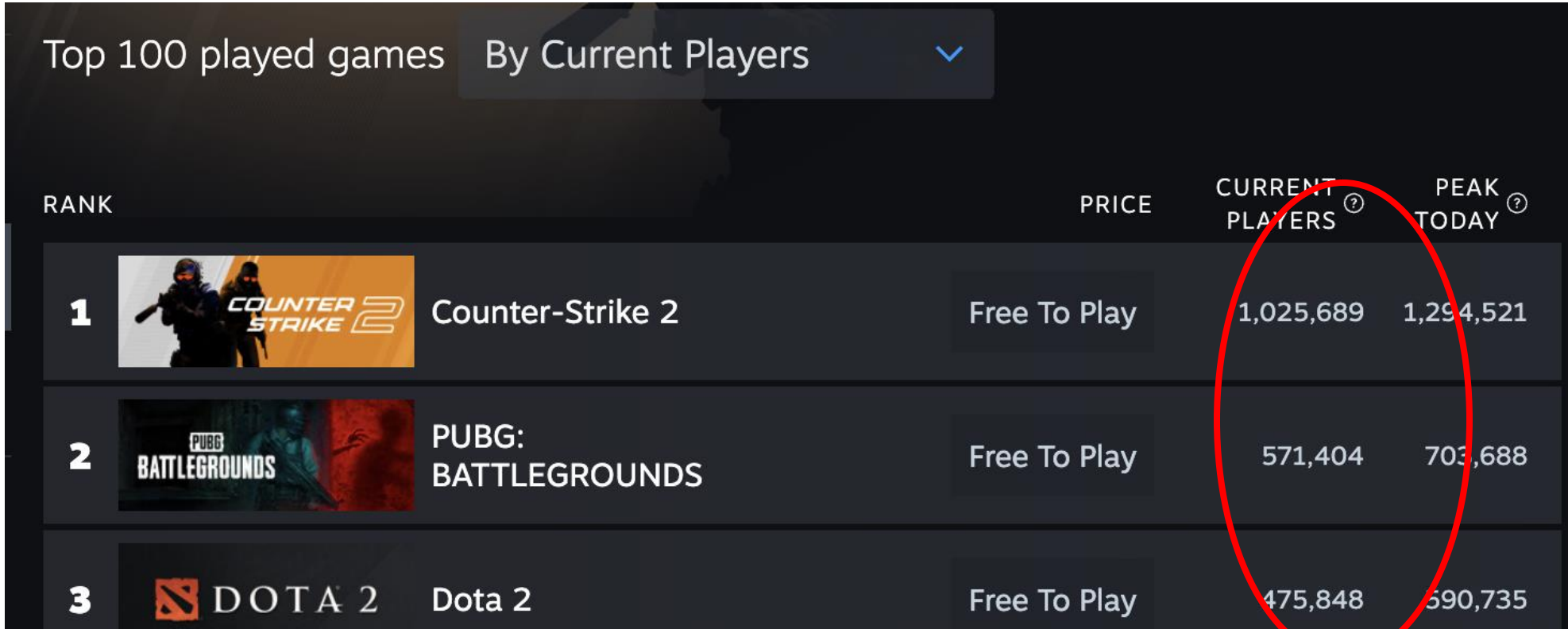


Bananen-Software

- Ursprünglich schlechter Ruf
- Jeder Nutzer rennt in dieselben Probleme

Nicht so bei SaaS

- Nutzer, die Probleme reporten, helfen allen
- Unendliche Ressourcen



Testen im Feld

- Nicht für jede Software im gleichen Maße praktikabel
- Aber: jede Software sollte im Feld beobachtet werden. Nicht nur durch Menschen.
- Hersteller benötigt Rückmeldung mit genügend Zusatzinformationen
- weitestgehend automatisch

Werkzeuge

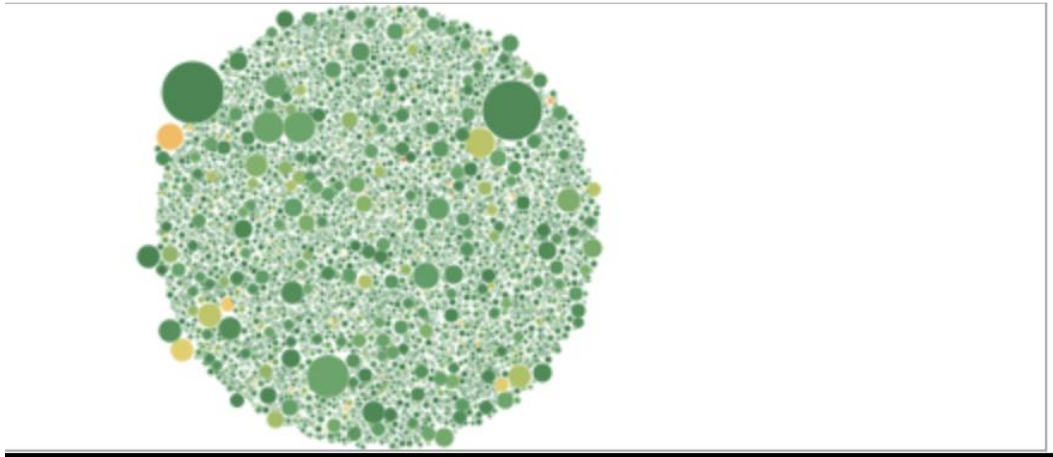
- Telemetry – Metriken
- Logging –Tracking aller Aktionen & Fehler.
- Fehlerreports von Nutzern.
- Nutzerzufriedenheit abfragen
- Monitoring aller Daten

Telemetry

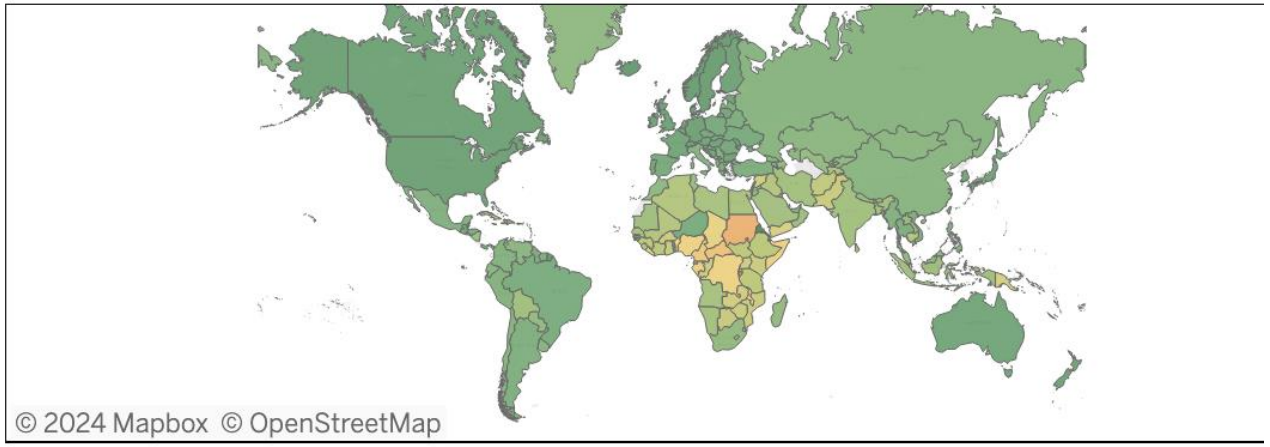
- Metriken in definierter, strukturierter Form - Events (CPU-Load, Network Latenz, Anzahl der Teilnehmer im Meeting, Feature Use).
- Typischerweise in aggregierter Form visualisiert.
- Als Flow für einzelne Sessions zur Analyse.
- Senderseitige Bibliothek zum versenden der Telemetry
- Speicher für Telemetry Events im Backend (z.B. Prometheus, InfluxDB, Elastic Search)
- Visualisierungstools (Grafana, Kibana).
- Alerting

Telemetry Example: Audio Quality

Quality by Customer Domain



Quality by Country



Details by Customer Domain

	Participants	VoIP Rx Quality/..	VoIP Tx Quality/..	PSTN Tx Quality..	Avg. VoIP Rx Lo..	Avg. PSTN Tx Lo..	Avg. VoIP Tx Lo..	Avg. VoIP Rx De..	Avg. VoIP Tx De..	VoIP Connection.
n	65,623	4.167	4.101	2.860	0.302	0.095	0.323	0.273	0.324	0.95
n	60,935	4.118	4.148	2.881	0.338	0.092	0.274	0.317	0.293	1.00
n	33,558	3.707	3.958	2.717	0.454	0.073	0.318	0.341	0.320	0.96
n	19,736	3.014	3.239		0.962	0.000	0.129	0.650	0.115	1.70
y	19,200	3.709	3.861		0.459		0.210	0.316	0.235	1.25
ir	17,375	3.718	3.645	2.734	0.549	0.104	0.186	0.444	0.201	1.11
t	15,188	2.569	2.695	2.683	0.704	0.150	0.176	0.471	0.140	2.51
n	14,320	3.532	3.776	2.884	0.637	0.048	0.247	0.503	0.254	1.15
..	13,466	4.082	4.100	2.833	0.199	0.115	0.227	0.200	0.239	0.62
n	12,188	3.796	3.999	2.867	0.449	0.055	0.216	0.349	0.198	0.98
	0K 50K 100K	0.0 0.5 1.0	0.0 0.5 1.0	0.0 0.5 1.0	0 2 4	0 2 4	0 2 4	0 2 4	0 2 4	0 10 20

Logging

- Optimal mit Struktur,evtl. binär, oft Freitextformat.
- Client und Server
- nicht nur Fehler loggen, sondern Abläufe
- Senderseitige Bibliothek
- Logstorage (Splunk, SumoLogic, S3, Loki) und Tool zur Analyse bzw. Visualisierung.
- Nur High level logs werden ständig übertragen.
- Im Problemfall Zugriff auf detailliertere Logs auf der Maschine
- Auffinden von unerwarteten Abläufen durch Exploration

Example: Logging Analysis Tool

Splunkfant - Session List - S3 based

Edit | Export | ...

Lists logfiles uploaded via S3 ingestion - click on a specific report to see the sessions contained in the log file - the last session is the one where the report was sent.

	ReportId	Email (User or Organizer) wildcard *	organizerAccountKey	User Issue Type
Last 7 days	*	*@goto.com	*	☐ Audio ☑ Video ☐ Other ☐ Screen Sharing

Submit | Hide Filters

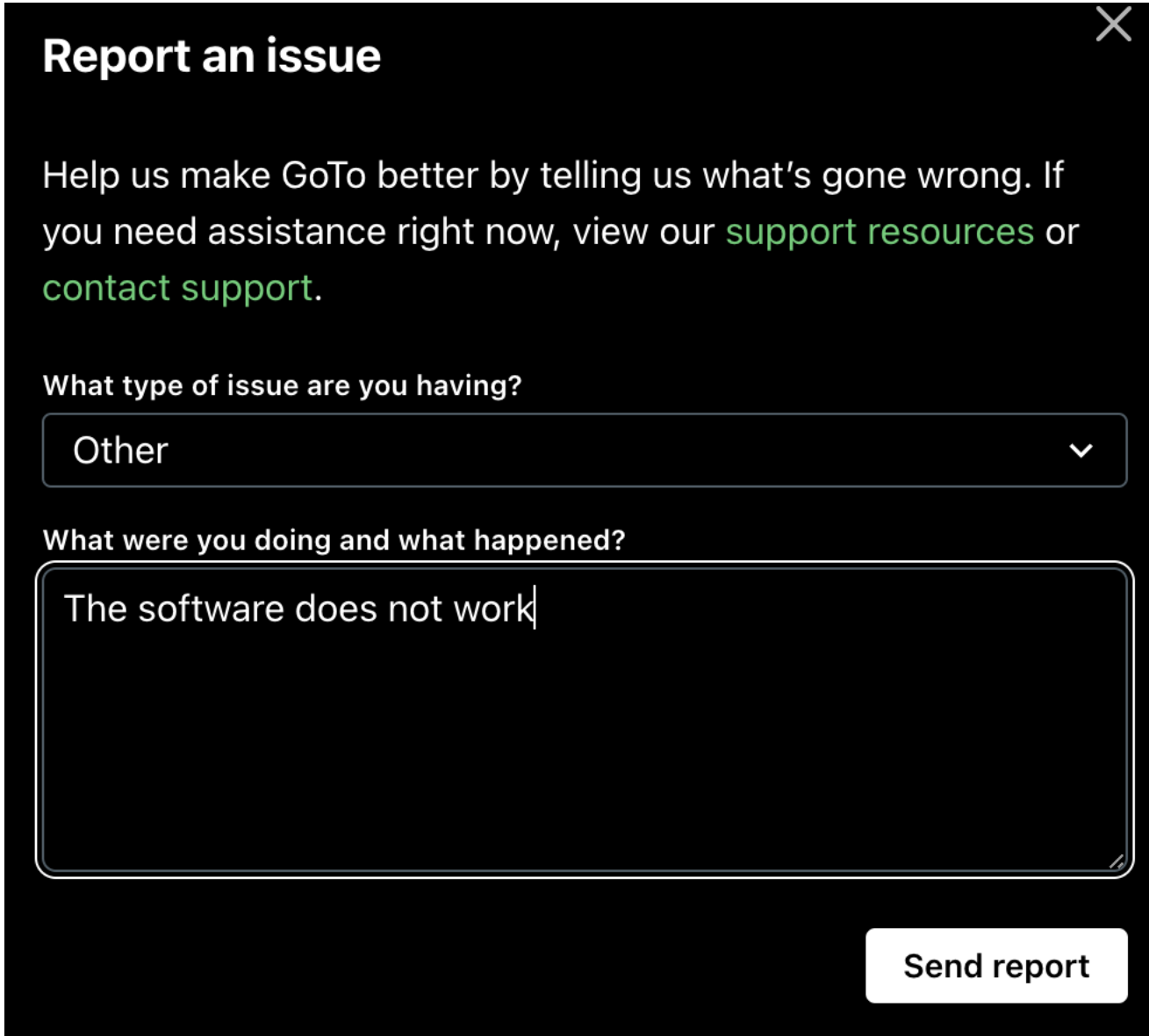
Reports

Click on the reportyou want to analyze - this will populate the panels below

userReportId	userComment	userName	userEmail	airStartTime	userIssueType	userType	sessionType	audioBridge	userMachineId	LOGSOURCE
A700-HEEI	all remote video ar frozen permanently	Robert Hess	robert.hess@goto.com	2024-10-22T13:10:44.550Z	video	internal	meeting	audio-ds-live-default-5c84fb7f78-hzbdg.k8s.goto-rtc.com	1ef6fff6-c3aa-5da7-9382-749f0b9e6dd8	s3://goto-issue-report-incoming/extracts/202410

Nutzerreports

- Teil der Anwendung,
- Nutzer beschreibt das Problem
- Kontext in Form von Logfiles, Identifikatoren wird automatisch angefügt



Kundenzufriedenheit ermitteln

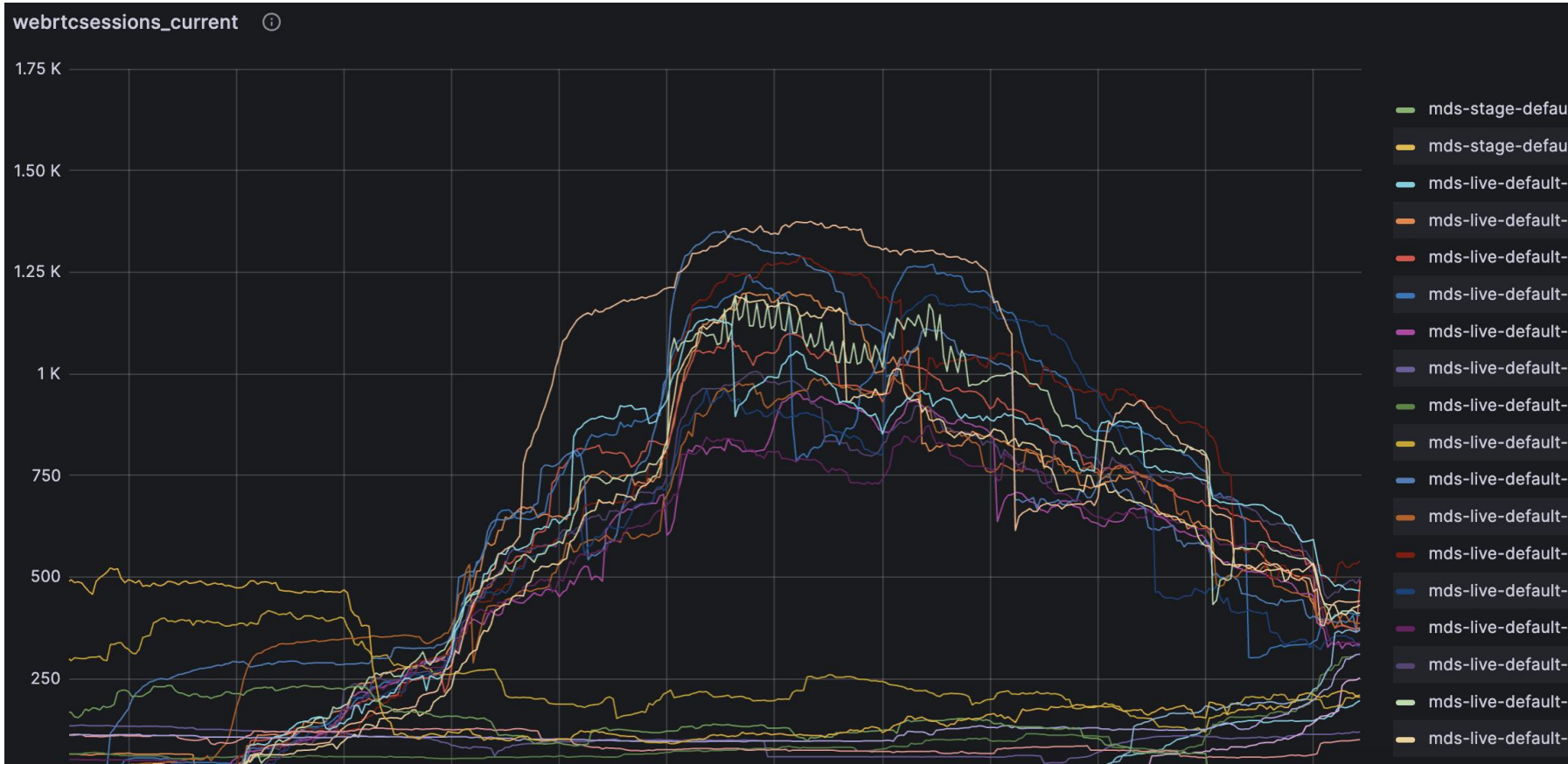
Auf einer Skale von 1 bis 10, wie hoch ist die Wahrscheinlichkeit, dass Sie dieses Produkt Ihren Freunden empfehlen würden.

Wichtiges Werkzeug um “weiche” Faktoren, wie Ergonomie und Feature-Vielfalt zu messen.

Z.B. Customer Engagement Score - CES.

Monitoring

- Visualisierung aller Daten
- Definition von Schwellwerten zur Alarmierung
- Vergleich mit historischen Daten



Zusammenfassung Feldtest

- Schrittweise Canary Release unter (automatisierter) Beobachtung der Metriken.
- Rollback, bei Grenzwerteüberschreitung, ggf. auch automatisch.
- In der Verantwortung der Software Teams, ebenso Rufbereitschaft (on-call).
- Zielgerichtete Experimente, um Erfolg bestimmter Veränderungen zu messen.
- Datenschutz berücksichtigen – viele dieser Daten sind personenbezogen und dürfen nur mit bestimmten Auflagen gesammelt werden.

Referenzen zum Weiterlesen

Qualities of old code

- https://security.googleblog.com/2024/09/eliminating-memory-safety-vulnerabilities-Android.html

Test Pyrmaid

- https://martinfowler.com/articles/practical-test-pyramid.html

On software re-writes

- https://www.joelonsoftware.com/2000/04/06/things-you-should-never-do-part-i/
- https://medium.com/@herbcaudill/lessons-from-6-software-rewrite-stories-635e4c8f7c22

Thanks

One day you'll understand, and everything will be more different than you can, at present, possibly hope to imagine.

(Thursday Next)