G+D
Currency Technology

Ricochet bugs

Why discovery in nightlies is too late to catch regressions

Stephan Schulz
G+D Currency Technology
TAV #51, Hamburg, 2.7.2025

In Nature there is Bugs .. and BUGS

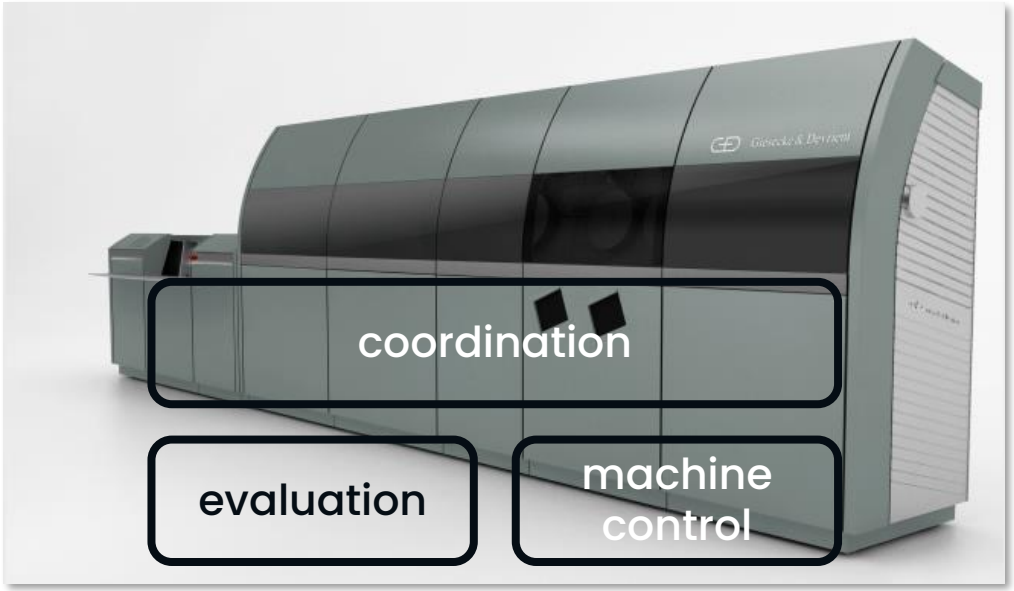
Lady Bug

- Cute … if it is just one
- Analogy: “cosmetic” SW bug

Giant Centipede

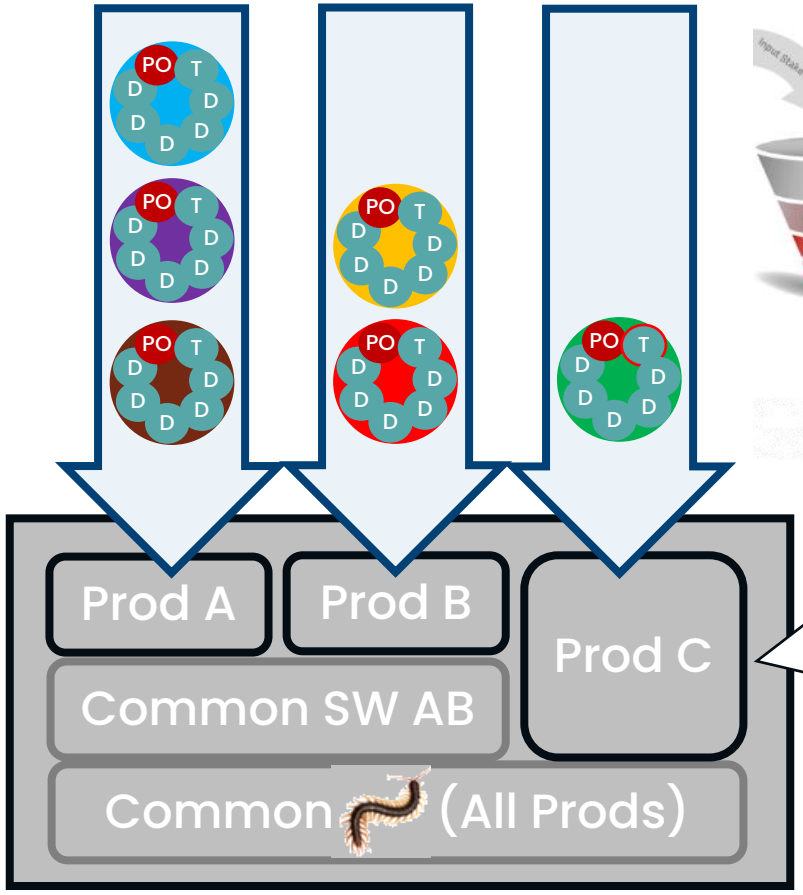
- Venomous .. eats other bugs
- Analogy: Ricochet SW bug aka “Querschläger”

What we test at G+D Currency Technology

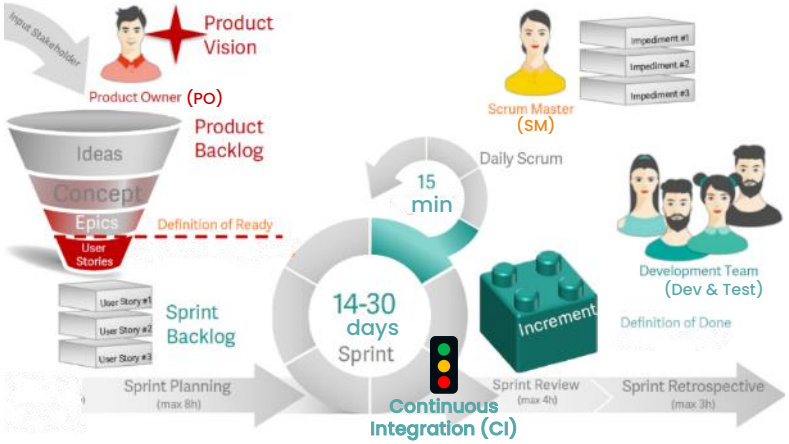


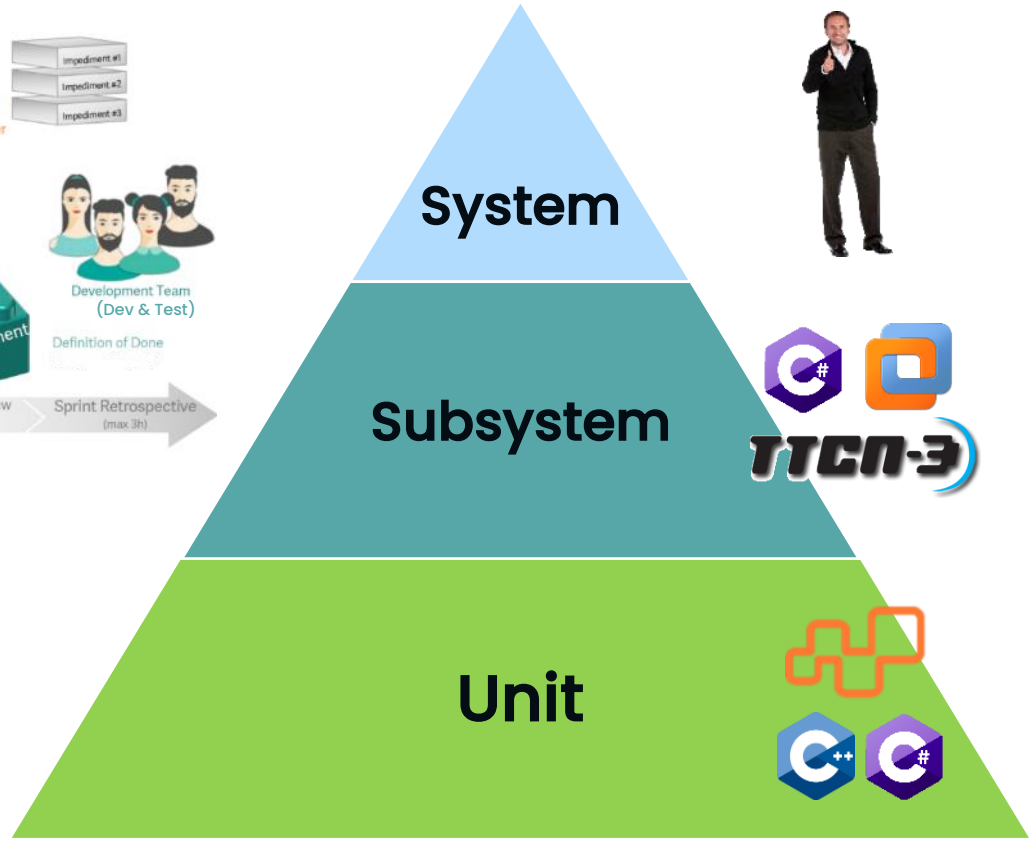
- Multiple products based on common product platform that are deployed worldwide in banknote printing, central banks, cash-in-transit centers, casinos, etc
- Each processes millions of banknotes/day **24h/7d** & is configurable for any currency
- Complex mixed HW/SW system including real-time SW, image processing, embedded control SW, highly concurrent processes, databases, etc.

How we develop & test our software



code base of millions LoC (incl legacy)



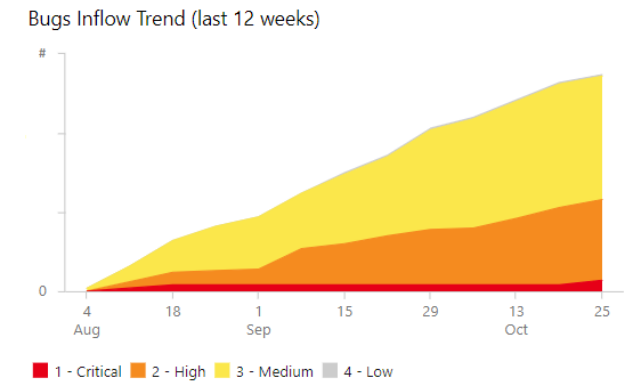


Ricochet Bugs – the Giant Centipede of the SW World!

„A bug introduced into a product via **platform code changes by another agile team** - often to fix a bug reported for *their* (other) product“

What makes them *so evil*?

- **Often** have significant impact on SUT behavior & **need to be fixed immediately**
- Usually **require help from 1-3 developers** from different teams to pinpoint
- **Generate unplanned work** (=spikes) from man days to **weeks** to analyze & fix
  - Take away capacity from *planned* bug fixes & feature/test development!
  - Create **negative impact on** own DEV **team morale**
- **Not well understood** in DEV (also management)
- ***Not* „just a test automation problem“**

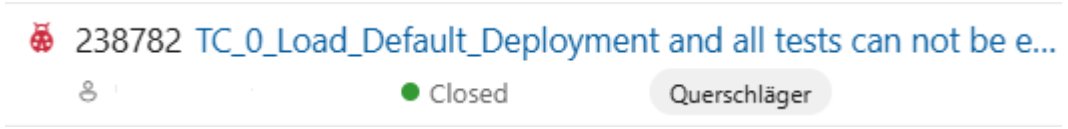


Some Interesting Observations from our Cases

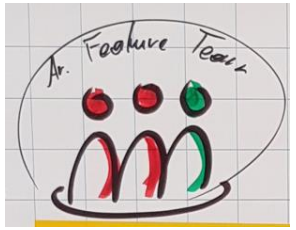
- Ricochet bugs occur **in waves** and their nature is **evolving**
- Their impact can vary **some to all nightlies** for one or all products
- Ricochet bugs **are rarely related** and **rarely simple to figure out**
  Note: Understanding dependencies within our code is not trivial!
- Most triggering developers were **thinking they did everything right** but **none of them tested or requested tests** prior to PR with *our* product
- A surprising percentage is triggered by (very) **senior** developers
- Effort spent on Ricochet bugs **in our DEV team** peaked at **1 FT (!) DEV**

So … how can we avoid Ricochet Bugs?

1. Create awareness & transparency



2. Early feature review with technical experts of all products

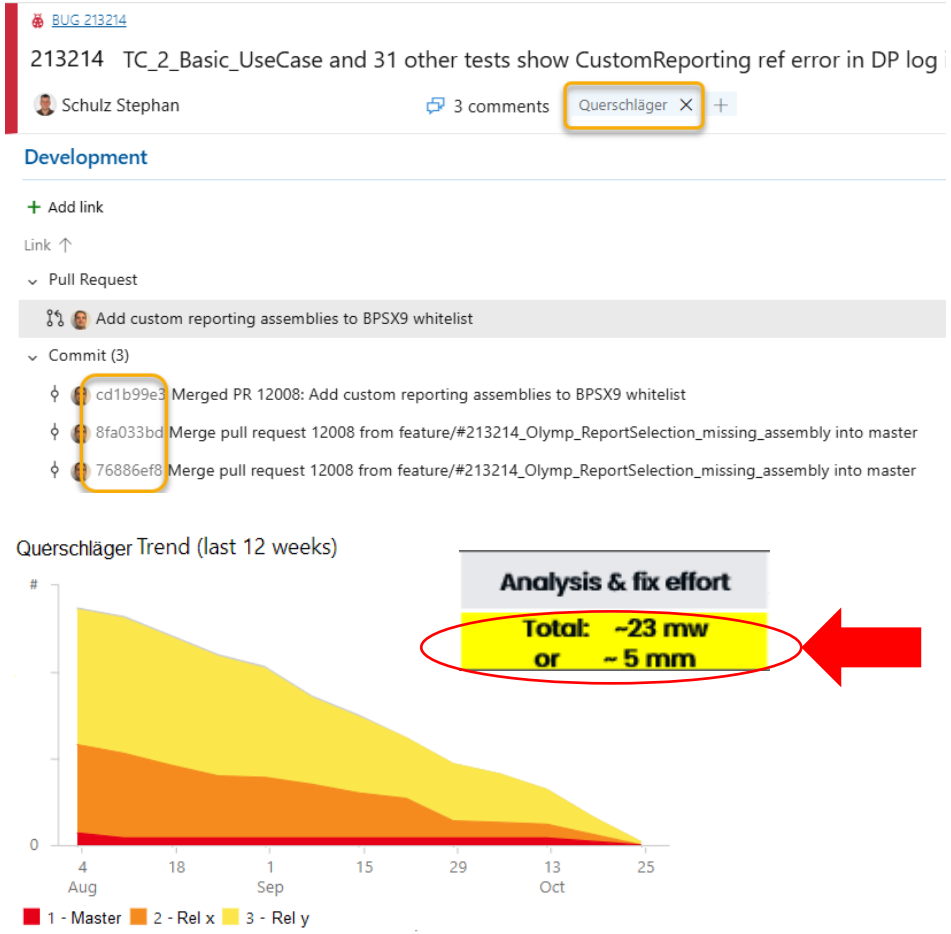


3. “Better” code reviews & system test automation in pull requests

4. Continuously improve our system test automation

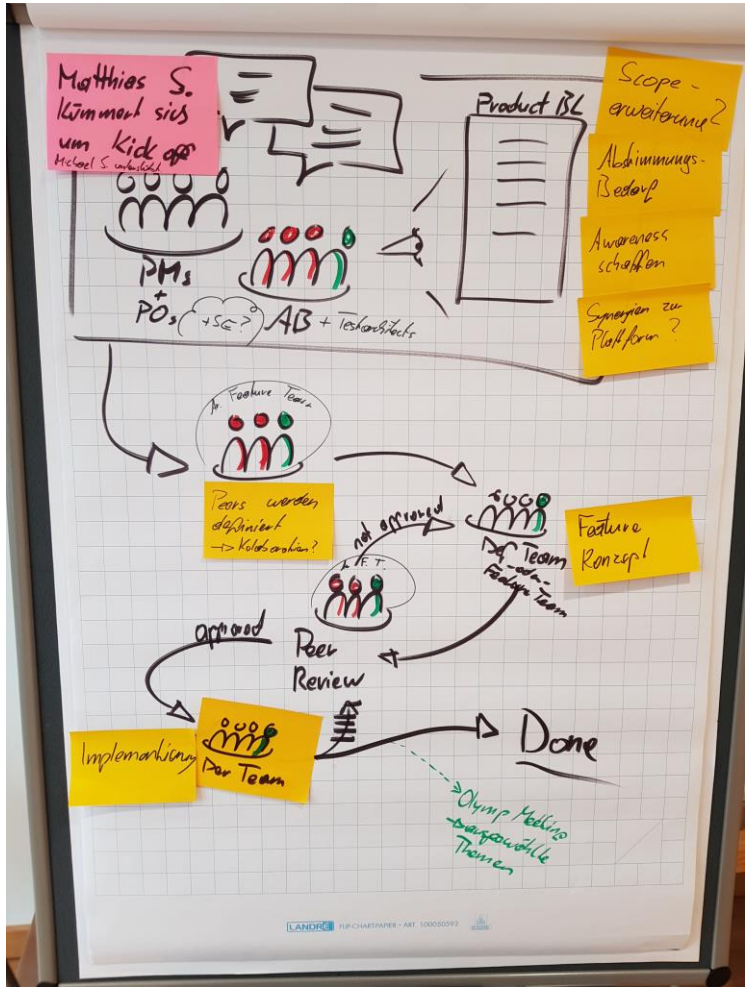
Create Awareness & Transparency!

- Track occurrences, e.g, **tag bug work items**, across all products and their releases
  - In particular when bugs are found by manual test, service, or customer (!)
- **Continuously analyze** causes **& improve** mechansims to find the next issue earlier
  - **Catalogue** problematic code sections
- Introduce metrics and KPIs
  - Create an understanding about **impact on product development** in management
  - Understand when/where we got better or when/why/where we got worse



Early Feature Review with Technical Experts!

- Involve [all] SW [test] architects in early product management feature reviews to identify features with a potential impact to the platform
- Create independent “feature teams” to review in a bit more detail with technical experts
- Record potential & known pitfalls for upcoming feature design & implementation
- Recommend automatic tests to run per product before pull requests



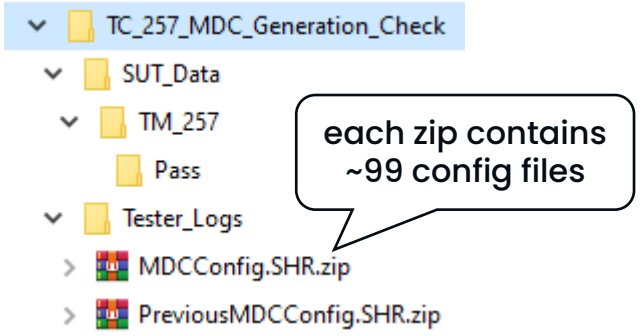
Better Reviews & Earlier System Test Automation!

- Manually run **system test automation on feature branches** (= BEFORE code goes to master, i.e., nightlies), to eliminate cross team analysis & fixing efforts 
- **Automatically select reviewers** (across different products) **for pull requests** based on changes to (known problematic) code sections 
- Make **system test automation a gating criteria for pull requests** with changes to known **problematic code sections** 
  - Focus on the (other) products that had already once issues here
- Later: Use of system test automation code coverage information to suggest relevant tests to execute (and help identify gaps in test coverage)

git version control has been a key technology here for making this work!

Continuously Improve our System Test Automation!

- Some fish will always escape our net but we want that number to approach zero!
  - Gradually make our net bigger and the holes in our net smaller!
- Many bugs slip our tests because they require a special SUT „configuration“
  - Often it is easy to **extend our system test automation** to cover them
  - Ensure „our“ user rights setup follows the one used „by most our customers“
- **Use system test automation also to just warn of changes** to (generated) SUT configuration files



In a brighter world …

80% of ricochet bugs will be found on “feature branches” (= never result in BUGS)!

19% of ricochet bugs will be found in test automation & fixed same day!

<1% of ricochet bugs will make it to the customer ☺!