# Playing With Your Project Data in Scrum Retrospectives

Christoph Matthies, Hasso Plattner Institute, University of Potsdam, Germany

## Scrum & Retrospectives

- Retrospectives: Agile's process improvement approach
- They should "start with the hard data" [1]
- However, programmers' beliefs primarily formed on personal experience rather than empirical project evidence [2]

Kniberg: "retro is the number-one-most-important thing in Scrum" [3]

### Prioritized Stakeholde Product Requirement Daily Backloc

### **Retrospective Games**

- Activities to structure Retros & encourage sharing impressions [1]
- Most popular resource lists 139 activities, 36 to gather data [4]

Only 2/36 proposed gather data activities take project data into account!

#### christoph.matthies@hpi.de



Paper C. Matthies, "Playing With Your Project Data in Scrum Retrospectives," in 42nd International Conference on Software Engineering Companion, ACM Press, 2020. doi: 10.1145/3377812.3382164

#### References

[1] E. Derby, D. Larsen. 2006. "Agile retrospectives: Making Good Teams Great." Pragmatic Bookshelf. [2] P. Devanbu, T. Zimmermann, C. Bird. 2016. "Belief & evidence in empirical software engineering." In Proceedings of ICSE 2016. [3] H. Kniberg. 2015. "Scrum and XP From the Trenches" (2nd ed.). C4Media. [4] C. Baldauf. 2018. "Retromat - Run great agile retrospectives!" Leanpub, Layton. [5] C. Souza, J. Froehlich, P. Dourish. 2005. "Seeking the Source: Software Source Code as a Social and Technical Artifact." In ACM SIGGROUP 2005.

# Software Project Data

# **Project Data Activities**

### Example: Progre

- Issue: Single
- Train team ir
- Estimate the
- Track contrib
  - git short

participants

Software engineers continuously document their work,

e.g. work item descriptions, test logs, and code commits. Employed development processes are "inscribed" into project artifacts [5] Provides evidence for problems, e.g. build failures, bug reports

Actively use (& play with) own project data in Retrospective meetings Learn from last iteration's project data for the next iteration Use tools that devs are already familiar with, e.g. git Devs are experts on their own data -> interpretable results! Start discussions on improvements based on team data

ess Check	Data Source	Example Measurement
dev commits all code	Version Control	Regular, small code commits
n VCS usage	Issue Tracker	Equal distribution of work items
improvement	Software Tests	Regularity of builds, failures
	Status Monitor	Accumulated downtime
butor count using	Comm. Tools	Communication patterns
log -sn wc -l	Code Analysis	Development of code coverage
	1.	

and their, own project data realities!

Contrast **mental process model** of Retrospective

