A Poor Man's Approach to Technical Debt

Marvin Kampf
Florian Wittmann

11. April 2014

Outline

Outline

Motivation

What is Technical Debt?

- A metaphor introduced by Ward Cunningham in 1992
- Describes the pending effort of maintenance that is needed to reach a certain quality level
- Is subject in many former works
- Necessary for statements:
  A model for quantifying and measuring technical debt

Motivation

Why another approach?

- Planning and issue tracker data from major German software manufacturer
- Easy and fast
- Comparison of different development states
- Analysis of technical debt and efficiency for the different stages

Outline

Two different states

Normal State

- Stages A, B1, B3 and C
- normal development process

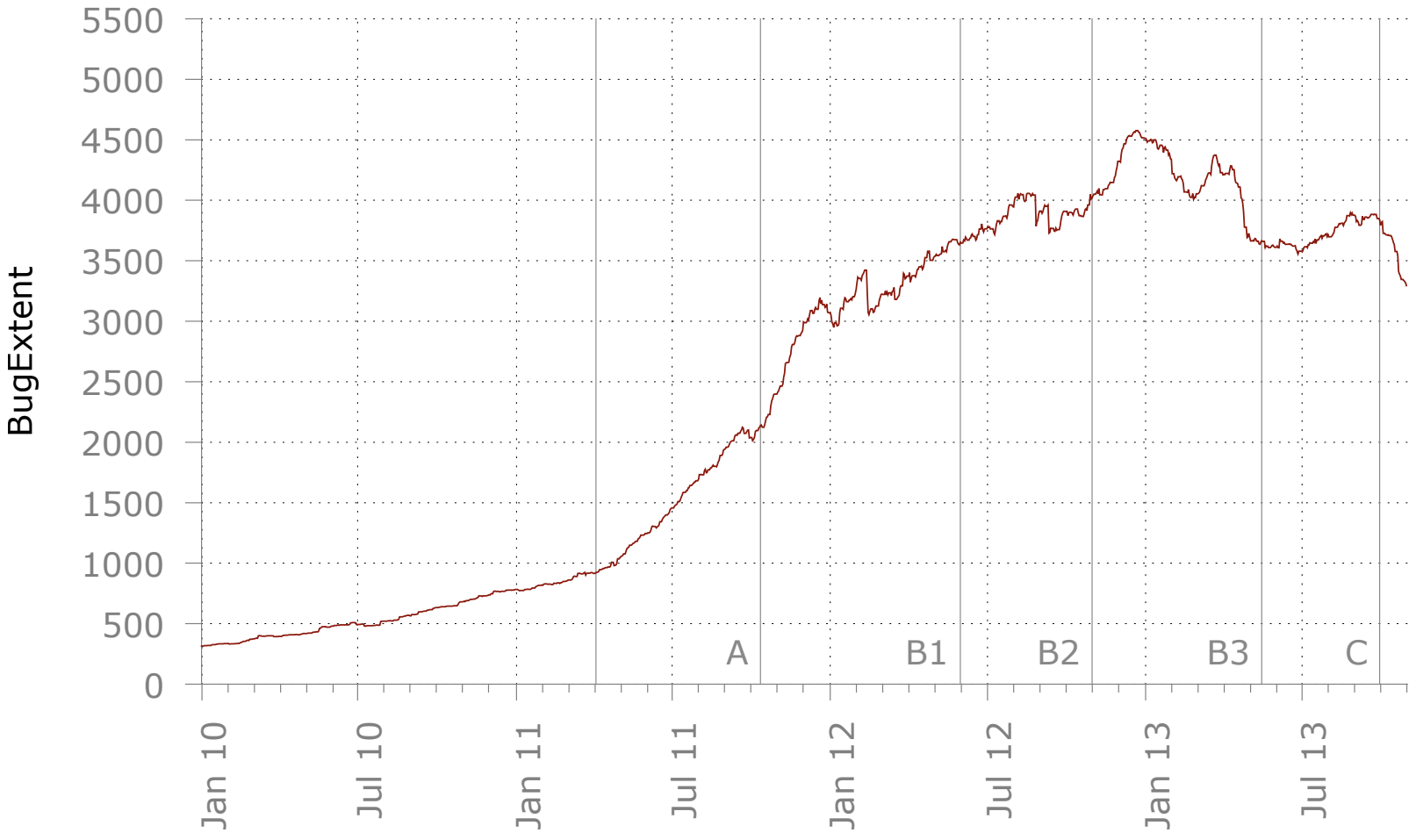
State of Emergency

- Stage B2
- focus on features

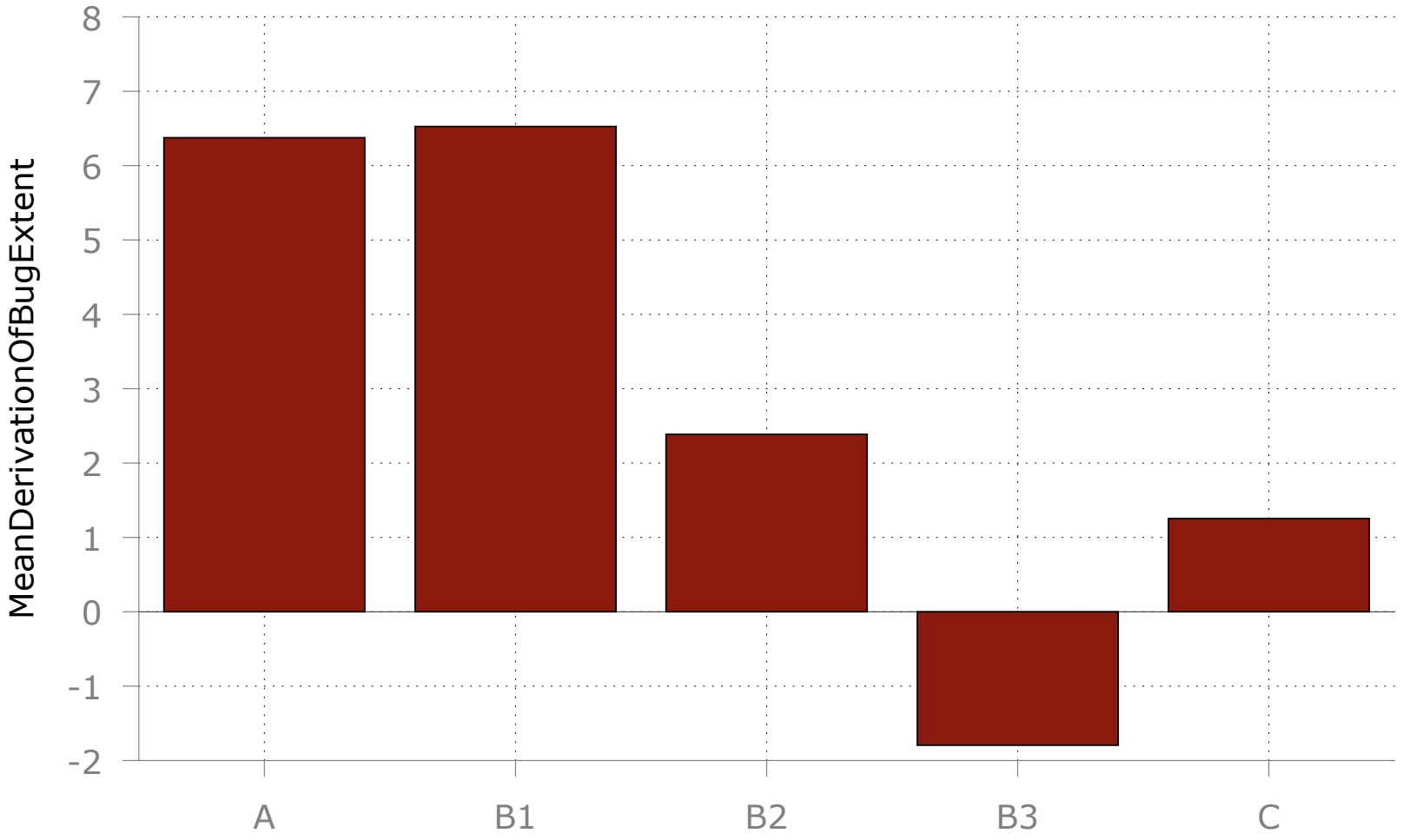
Provided Data - Important fields

IssueType	The type of a certain issue (*User Story*, *Defect* or *Feature*).
EffortSec	The real effort that was spent to implement a feature, user story or to fix a defect, depending on the issue type.
IssueKey	The issue's unique key to identify it.
FeatureKey	The parent ID. If a user story is assigned to a parent feature, this feature's IssueKey is stored here.
CreationDate	The date the issue was created.
ComplDate	The date the issue has been completed.

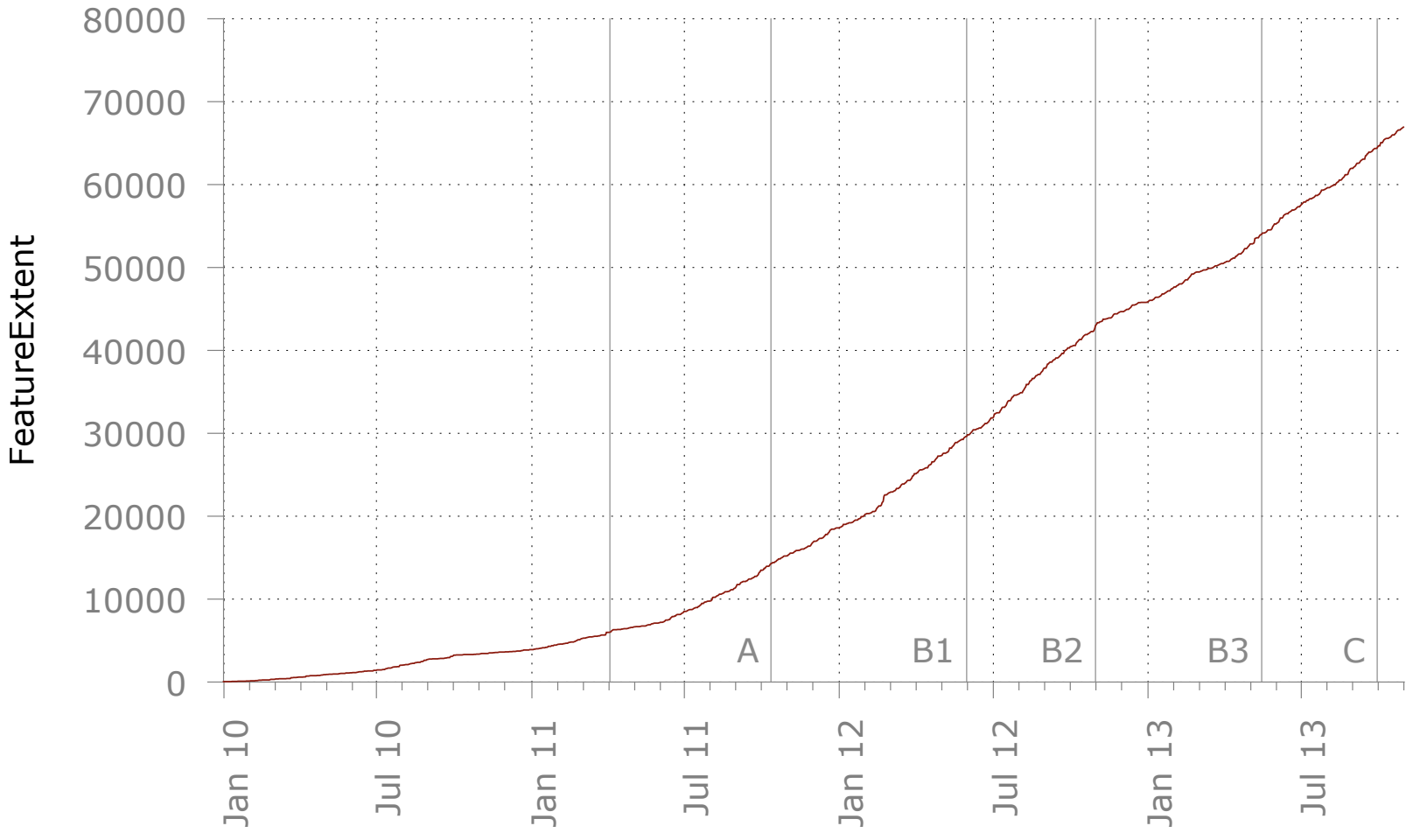
Bug Extent


BugExtent
5500
5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
A
B1
B2
B3
C
Jan 10
Jul 10
Jan 11
Jul 11
Jan 12
Jul 12
Jan 13
Jul 13

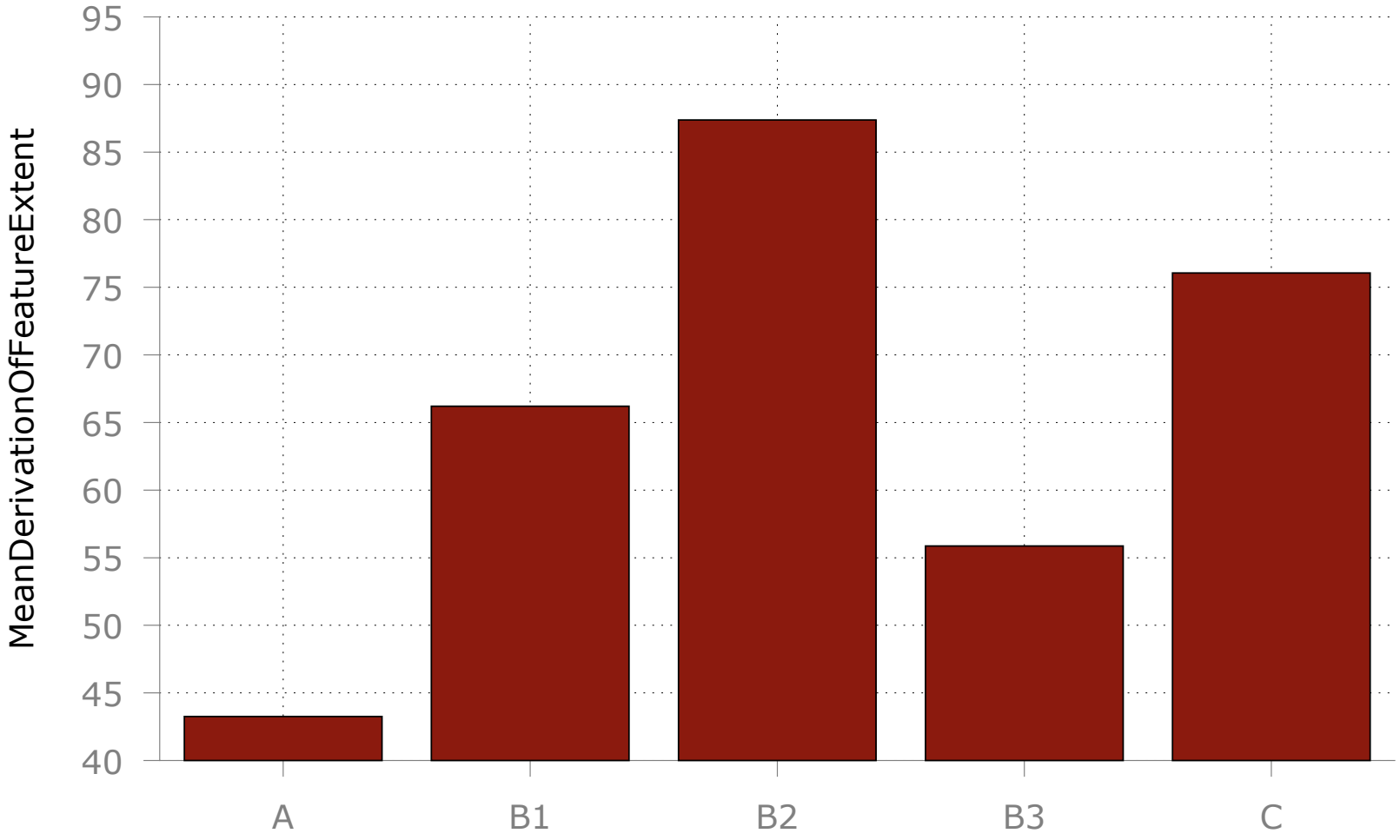
Mean derivate of Bug Extent



Feature Extent



Mean derivate of Feature Extent



Outline

Measuring Technical Debt

Bug Count

$$d(t) := n_{bug}(t)$$

Bug Extent

$$d(t) := \sum_{i}^{n_{bug}(t)} e_{bug,i} \cdot s_{bug,i}$$

e_{bug} Real effort in days to fix a bug

s_{bug} Severity of a bug

Quality Level

$$QualityLevel = \frac{Proceeds}{Deficiency}$$

Feature Extent

$$f(t) := \sum_{i}^{n_{feat}(t)} e_{feat,i} \cdot s_{feat,i}$$

$n_{feat}(t)$ All completed features at time t

e_{feat} Real effort in days

s_{feat} Importance of a feature

Quality Level

$$q(t) := \frac{f(t)}{d(t)}$$

Efficiency

Efficiency

$$Eff(\Delta t) := \frac{e_{push}}{c_{team} \cdot \Delta t}$$

e_{push} Effort for all completed features

c_{team} Costs for team

Normed Efficiency

$$\overline{Eff}(\Delta t) := Eff \cdot (1.0 + \frac{q(t_1) - q(t_0)}{|q(t_{end}) - q(t_{start})|})$$

t_0 Start of the current stage

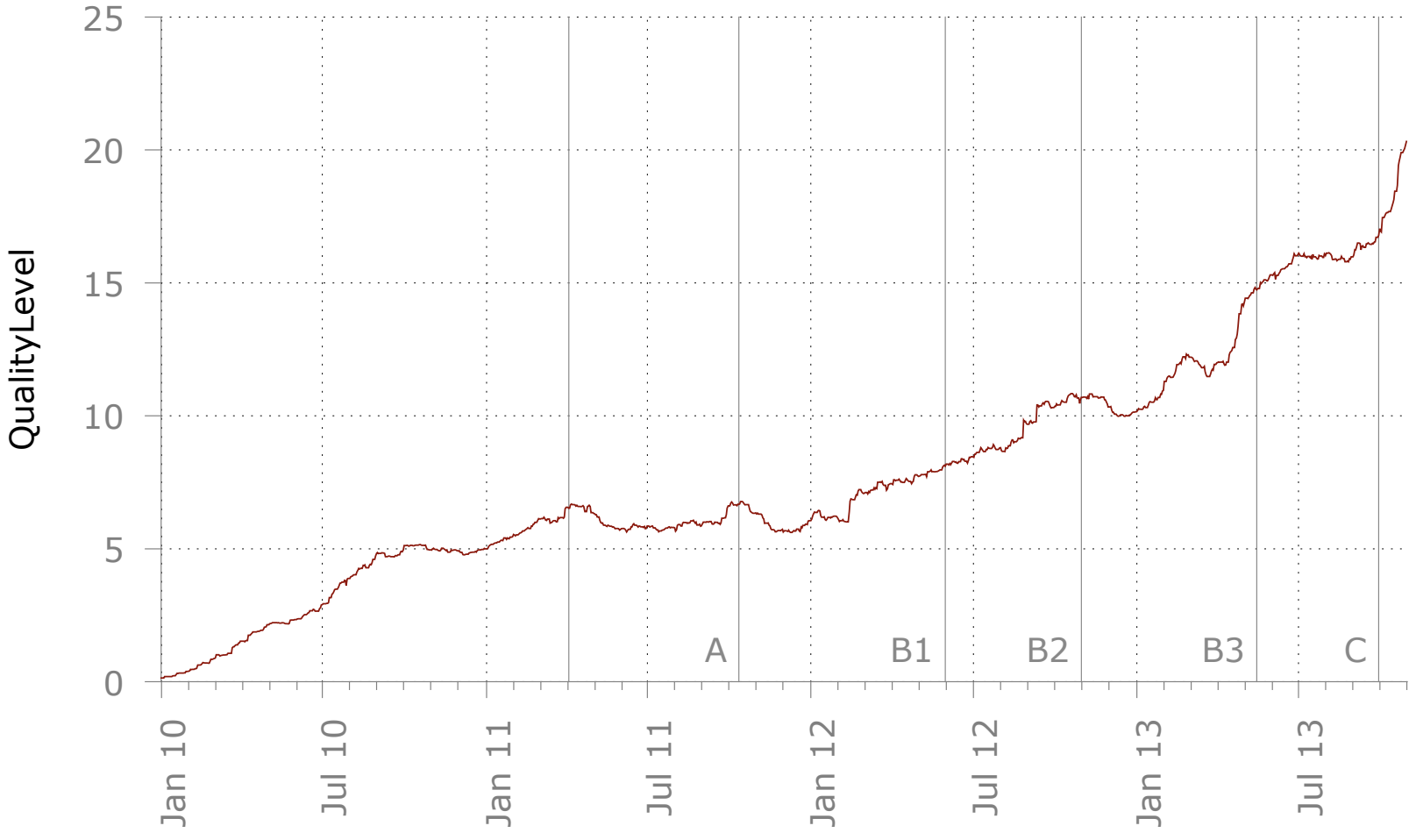
t_1 End of the current stage

t_{start} Global start across all stages

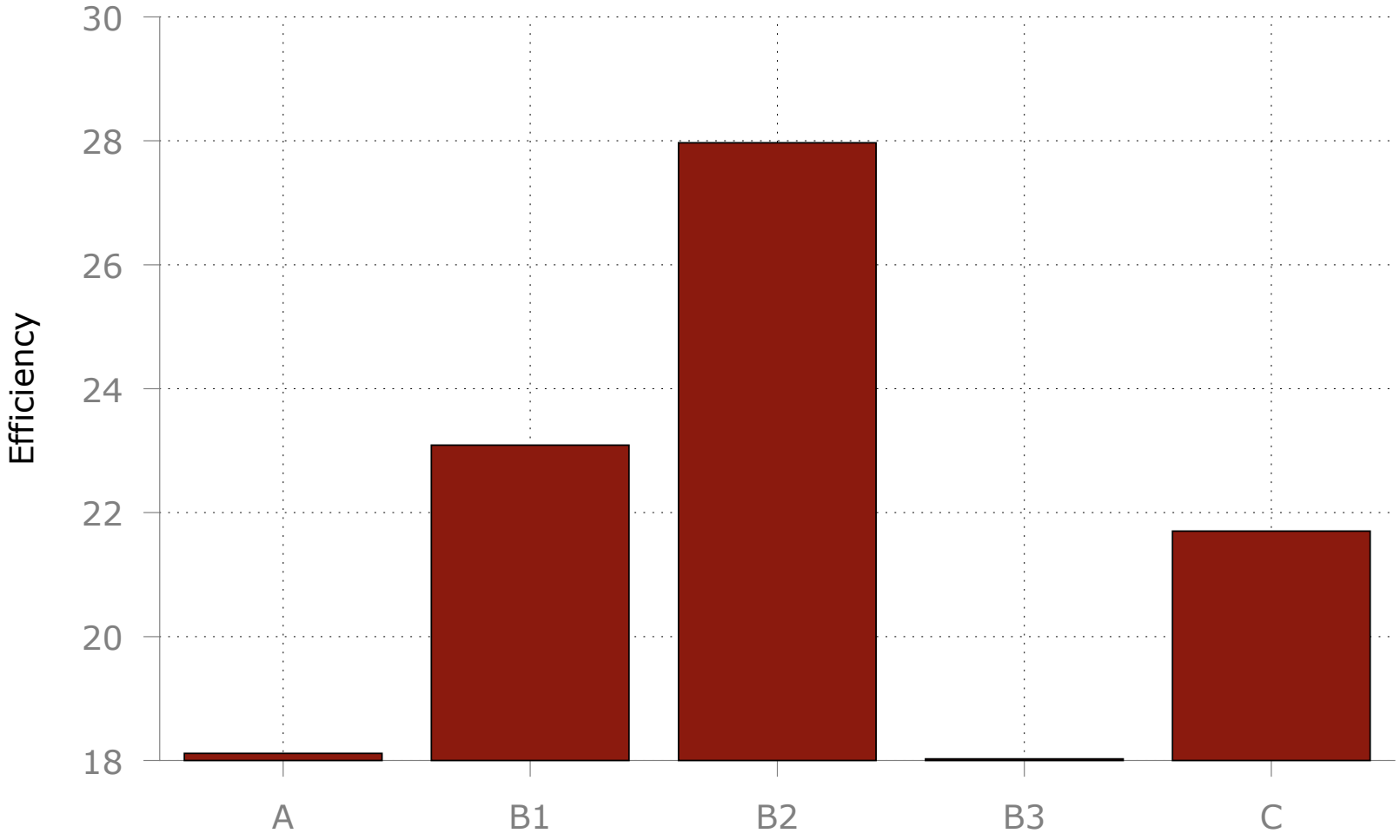
t_{end} Global end across all stages

Outline

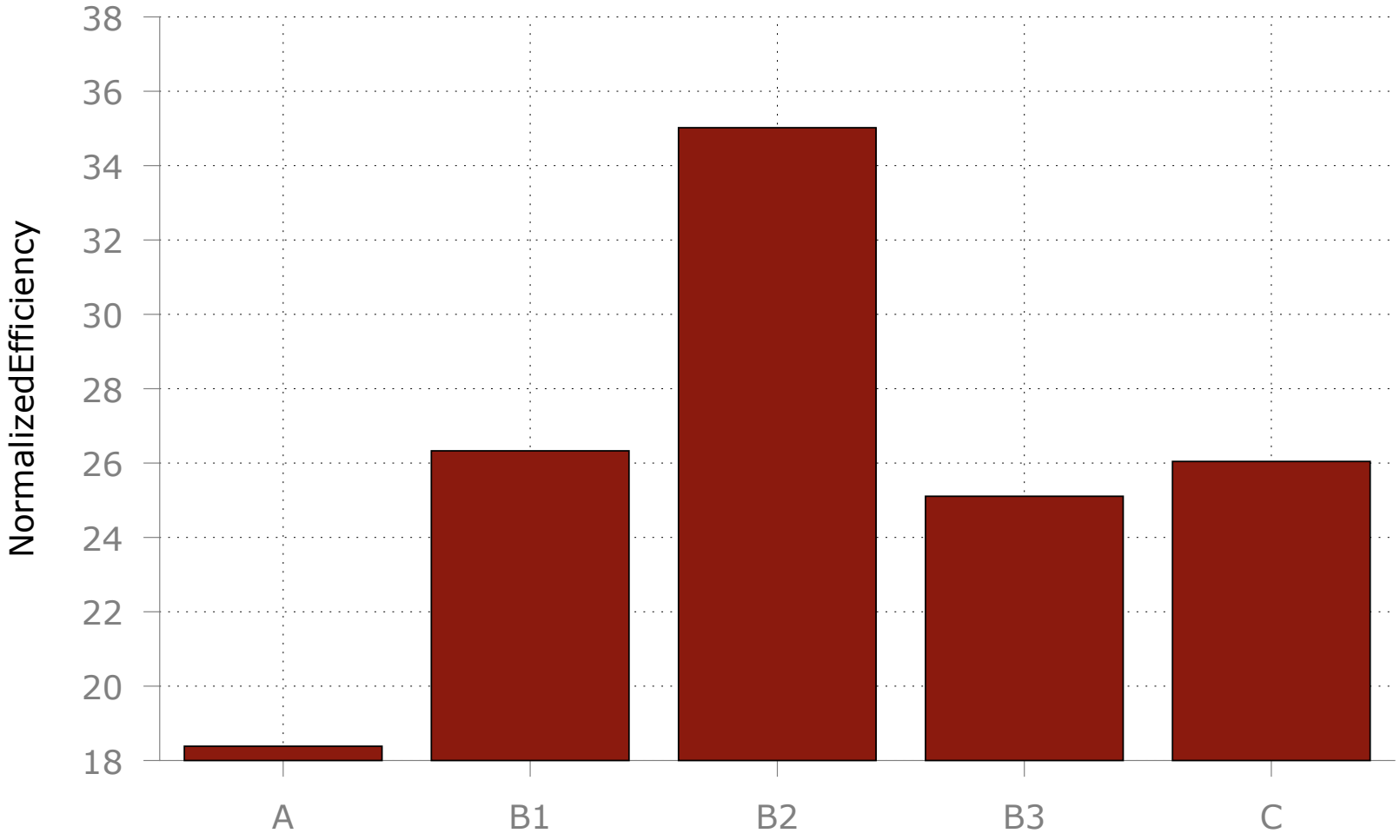
Quality Level



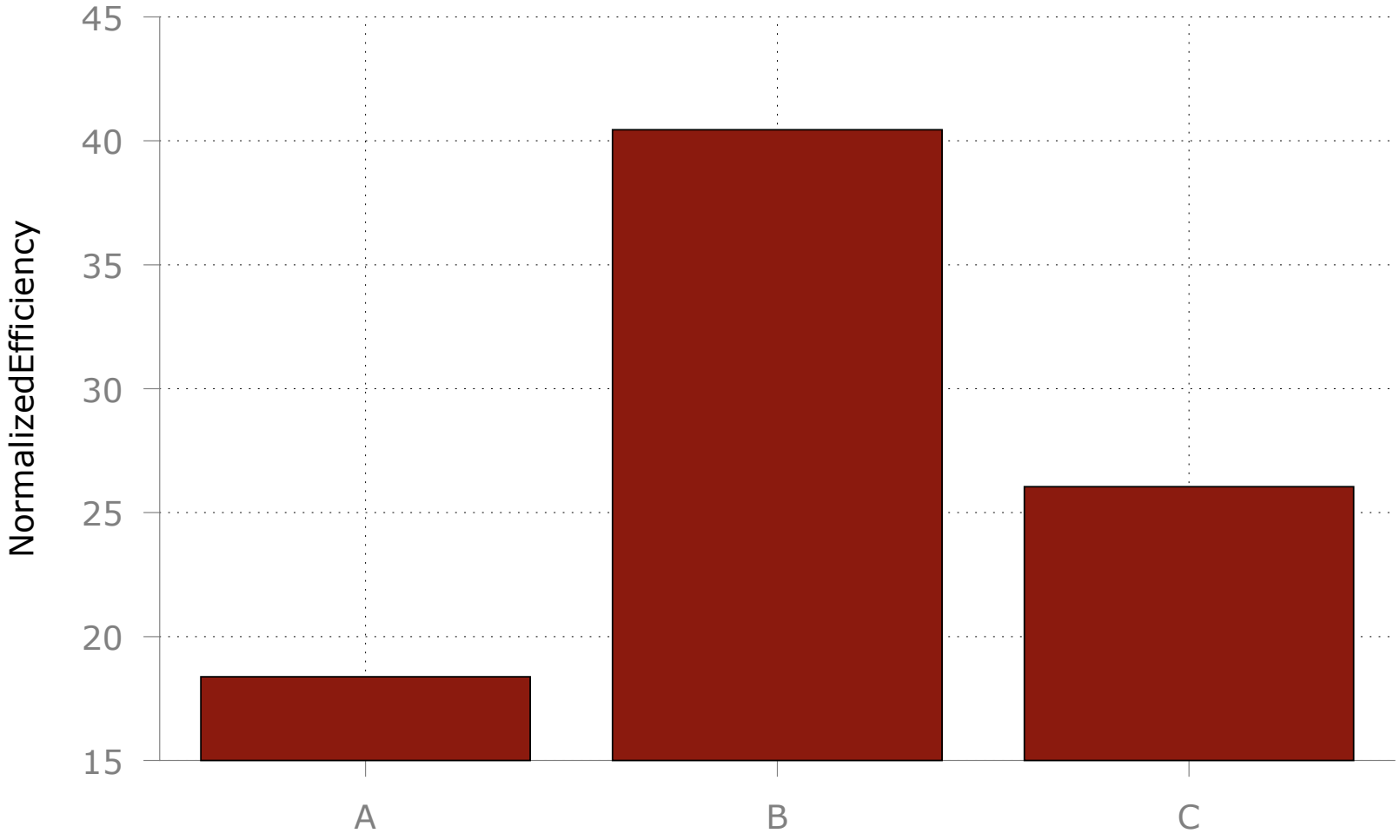
Efficiency



Normalized Efficiency



Normalized Efficiency



Outline

Discussion

- Bug Extent is much more significant than Bug Count
- Feature Extent is increasing (In B2 it is faster rising than in the other stages)
- The resulting quality level is rising shakily
- In B2 the absolute efficiency was the highest
- Even after normalizing with the quality level B2 has still the best efficiency

Limits

- In our example we assumed same severity for all bugs and all features.
- Effort for closing bugs calculated for last stage, even if bug was fixed during more than one stage