Anleitung Continuous Improvement über GitHub CLI

1. Throughput siehe (TM - Praktischer Teil - GitHub CLI.pdf)
2. Cycle-/Leadtime

```
Cycle Time & PR Lead Time:

# ---- Einstellungen für Zeitformat aus der GitHub-API ----
$fmt   = 'yyyy-MM-ddTHH:mm:ssZ'
$cult  = [System.Globalization.CultureInfo]::InvariantCulture
$style = [System.Globalization.DateTimeStyles]::AssumeUniversal

# ---- 1) PRs holen (alle Zustände) ----
$prs = gh pr list --state all --json number,title,createdAt,mergedAt,url |
       ConvertFrom-Json

# ---- 2) Für jede PR: FirstCommit ermitteln, Zeiten berechnen ----
$rows = foreach ($pr in $prs) {
  # Grunddaten parsen
  $created = $null
  if ($pr.createdAt) {
    $created = [datetime]::ParseExact(($pr.createdAt | Select-Object -First 1), $fmt, $cult,
$style)
  }

  $merged = $null
  if ($pr.mergedAt) {
    $merged = [datetime]::ParseExact(($pr.mergedAt  | Select-Object -First 1), $fmt, $cult,
$style)
  }

  # Ersten Commit des PRs holen (falls vorhanden)
  $firstCommit = $null
  try {
    $v = gh pr view $($pr.number) --json commits | ConvertFrom-Json
    # Die Commits kommen meist als Array mit property "committedDate"
    $firstCommitRaw = ($v.commits | Sort-Object committedDate | Select-Object -First
1).committedDate
    if ($firstCommitRaw) {
      $firstCommit = [datetime]::ParseExact(($firstCommitRaw | Select-Object -First 1),
$fmt, $cult, $style)
    }
  } catch {
    # PR hat evtl. (noch) keine Commits / keine Rechte / Draft etc.
    $firstCommit = $null
  }

  # Kennzahlen berechnen (nur wenn gemerged)
  $leadH  = $null; $leadD  = $null
  $cycleH = $null; $cycleD = $null

  if ($merged -ne $null -and $created -ne $null) {
    $leadH = [math]::Round(($merged - $created).TotalHours, 2)
    $leadD = [math]::Round(($merged - $created).TotalDays,  2)
  }

  if ($merged -ne $null -and $firstCommit -ne $null) {
    $cycleH = [math]::Round(($merged - $firstCommit).TotalHours, 2)
    $cycleD = [math]::Round(($merged - $firstCommit).TotalDays,  2)
  }

  [pscustomobject]@{
    Number        = $pr.number
    Title         = $pr.title
    Opened        = $created
    FirstCommit   = $firstCommit
    Merged        = $merged
```

```
    'LeadTime(h)' = if ($leadH  -ne $null) { $leadH  } elseif ($merged) { 'n/a' } else {
'open' }
    'LeadTime(d)' = if ($leadD  -ne $null) { $leadD  } elseif ($merged) { 'n/a' } else {
'open' }
    'CycleTime(h)'= if ($cycleH -ne $null) { $cycleH } elseif ($merged) { 'n/a' } else {
'open' }
    'CycleTime(d)'= if ($cycleD -ne $null) { $cycleD } elseif ($merged) { 'n/a' } else {
'open' }
    URL           = $pr.url
  }
}

# ---- 3) Durchschnittswerte (nur numerische Zeilen) ----
$avgLeadH  = [math]::Round( ( $rows | Where-Object { $_.'LeadTime(h)'  -is [double] }
| Measure-Object 'LeadTime(h)'  -Average ).Average, 2)
$avgLeadD  = [math]::Round( ( $rows | Where-Object { $_.'LeadTime(d)'  -is [double] }
| Measure-Object 'LeadTime(d)'  -Average ).Average, 2)
$avgCycleH = [math]::Round( ( $rows | Where-Object { $_.'CycleTime(h)' -is [double] }
| Measure-Object 'CycleTime(h)' -Average ).Average, 2)
$avgCycleD = [math]::Round( ( $rows | Where-Object { $_.'CycleTime(d)' -is [double] }
| Measure-Object 'CycleTime(d)' -Average ).Average, 2)

# ---- 4) Durchschnittswerte als Extra-Spalten anhängen ----
$rows | ForEach-Object {
  $_ | Add-Member -NotePropertyName 'AvgLead(h)'  -NotePropertyValue $avgLeadH
  $_ | Add-Member -NotePropertyName 'AvgLead(d)'  -NotePropertyValue $avgLeadD
  $_ | Add-Member -NotePropertyName 'AvgCycle(h)' -NotePropertyValue $avgCycleH
  $_ | Add-Member -NotePropertyName 'AvgCycle(d)' -NotePropertyValue $avgCycleD
}

# ---- 5) Ausgabe ----
$rows | Format-Table -AutoSize
```

3. CI Success Rate

CI Success Rate.ps1

```
$Repo = "HSLU-Exercise/scope-your-project-gruppe-11"

$runs = gh run list -R $Repo --limit=200 --json conclusion | ConvertFrom-Json
$valid = $runs | Where-Object { $_.conclusion }
$total = $valid.Count
$success = ($valid | Where-Object { $_.conclusion -eq 'success' }).Count

if ($total -gt 0) {
  $rate = [math]::Round(($success / $total) * 100, 2)
  "CI Success Rate: $rate %  ($success von $total Runs erfolgreich)"
} else {
  "Keine abgeschlossenen Runs gefunden."
}
```

```
mester\IT Project Basics> powershell -ExecutionPolicy Bypass -File '.\CI Success Rate.ps1'
CI Success Rate: 16.5 %  (17 von 103 Runs erfolgreich)
```