Benchmark

# System software maintenance

Steps to benchmark system software

**1. Define Your Goals:**

* What aspects of maintenance do you want to benchmark? Common areas include:
  + **Efficiency:** Time taken to resolve issues, code changes per unit time.
  + **Cost:** Maintenance costs as a percentage of total software budget.
  + **Quality:** Number of defects identified during maintenance, user satisfaction with maintenance responsiveness.

**2. Choose Your Benchmarking Method:**

* **Internal Benchmarking:** Compare different teams or projects within your company. Useful for identifying internal performance variations.
* **Competitive Benchmarking:** Compare your performance against direct competitors. Requires gathering data, which might be challenging.
* **Industry Benchmarking:** Compare against industry averages or best practices published by research firms or industry associations.

**3. Identify Relevant Metrics:**

Select quantifiable metrics that align with your chosen goals. Here are some examples:

* **Mean Time to Resolution (MTTR):** Average time taken to fix a reported issue.
* **Number of Code Changes per Developer per Month:** Measures developer productivity.
* **Cost per Defect Fixed:** Tracks efficiency of maintenance efforts.
* **Customer Satisfaction Score:** Gauges user perception of maintenance responsiveness.

**4. Gather Data:**

* Utilize existing data sources like issue tracking systems, code repositories, and financial records.
* Conduct surveys or interviews with developers and end-users to gather qualitative feedback.
* If using external benchmarks, ensure data sources are reputable and relevant to your industry.

**5. Analyze and Interpret Results:**

* Compare your metrics against chosen benchmarks. Identify areas where your company excels or falls short.
* Consider factors that might influence your results, such as software complexity or team size.

**6. Develop Improvement Strategies:**

* Based on the analysis, develop strategies to address identified weaknesses. This could involve:
  + Streamlining maintenance processes.
  + Implementing new tools or technologies.
  + Optimizing resource allocation for maintenance activities.

**7. Continuously Monitor and Refine:**

* Regularly track your chosen metrics to monitor progress towards improvement goals.
* Refine your benchmarking strategy over time as your company's needs and software portfolio evolve.

By effectively benchmarking your system software maintenance, you gain valuable insights into your team's efficiency, cost management, and overall effectiveness. This data-driven approach allows you to continuously improve your maintenance practices, ultimately leading to a more reliable, cost-effective, and user-friendly software experience.

# Benchmarking Criteria

**1. Efficiency:**

* **Mean Time to Resolution (MTTR):** This measures the average time taken to resolve a reported issue. Lower MTTR indicates faster issue resolution and improved system uptime. Define acceptable MTTR targets based on issue severity levels (critical, high, medium, low).

| time/hr | <1 | >5 | >24 | >72 | >120 |
| --- | --- | --- | --- | --- | --- |
| points | 5 | 4 | 3 | 2 | 1 |

* **Code Changes per Developer per Month:** This metric reflects developer productivity in maintenance tasks. Balance high numbers with code quality and maintainability.
  + <5 (5 points)
  + <15 (4 points)
  + <20 (3 points)
  + <50 (2 points)
  + >75 (1 point)
* **First Fix Rate:** This measures the percentage of issues resolved on the first attempt. A higher rate reduces rework and improves efficiency.
  + <=100 (5 points)
  + <=75 (4 points)
  + <=50 (3 points)
  + <=40 (2 points)
  + <25 (1 point)

**2. Cost:**

* **Cost per Defect Fixed:** This tracks the efficiency of maintenance efforts. Ideally, this cost should decrease over time as processes improve.
  + Criteria depends on the work
* **Maintenance Cost as a Percentage of Total Software Budget:** This provides a high-level view of maintenance resource allocation. Aim for a balance between sufficient resources and cost-effectiveness.
  + Criteria depends on the work

**3. Quality:**

* **Number of Defects Identified During Maintenance:** Lower defect introduction during maintenance activities signifies better code quality and attention to detail.
  + Critical
    - >5 (10 points)
    - <10 (5 points)
    - <15 (3 points)
    - <20 (1 point)
  + Major
    - >5 (10 points)
    - <10 (5 points)
    - <15 (3 points)
    - <20 (1 point)
  + Medium
    - >5 (10 points)
    - <10 (5 points)
    - <15 (3 points)
    - <20 (1 point)
  + Cosmetic
    - >5 (10 points)
    - <10 (5 points)
    - <15 (3 points)
    - <20 (1 point)

**4. User Satisfaction:**

* **Customer Satisfaction Score:** Conduct surveys or interviews to gauge user perception of maintenance responsiveness, communication, and overall effectiveness.
  + High (10 points)
  + Medium (5 points)
  + Low (1 point)
* **Number of Support Tickets Reopened:** Tracks the frequency of issues recurring after being supposedly fixed. A low number indicates effective resolution.
  + >5 (10 points)
  + >10 (5 points)
  + >20 (1 point)