40+ Metrics for Software Teams

The following listing is intended as a starting point for conversation and discussion. Choose one or two that make sense for your team / organization / experiment and add them to your current dashboard. Rinse and repeat over time.

People/Team: Human Elements

Reveals issues that impact a team's engagement

- Team health and well-being
- Team / manager / org eNPS
- % of time w/o interruptions (fire fights)
- Trust between leadership and team
- Learning log
- Overtime hours
- New employee setup time
- Team tenure / manager half-life
- Mean time to reorg
- Whole team contribution
- Transparency (access to data, customers, learning)
- Employee silence vs voice
- Number of experiments / changes

Process Health Metrics

Day-to-day delivery activities and process changes

- Cycle time
- Cumulative flow diagrams
- Control charts
- Number of experiments performed
- Number of improvements made to process
- Flow efficiency
- "Concept to cash" delivery lead time
- Batch size
- Successful iteration completion
- Escaped defect resolution time
- Time thievery

Release Metrics

Identifying impediments to continuous delivery

- Deployment frequency
- Release success rate
- Failed build frequency
- Mean time to restore (MTTR)
- Release time
- Cost per release
- Escaped defects
- Release net promoter score
- Release adoption / install rate

Product Development Metrics

Measure alignment of product features to user needs

- Customer value delivered
- Risk burn down
- Value stream mapping
- Sales velocity
- Product net promoter score (NPS)
- User analytics / DAU / MAU
- Backlog health index
- Number of validated business-level hypotheses developed
- Number of times a week you talk to an actual customer
- Ratio of implemented to non-implemented customer-driven changes

Technical/Code Metrics

Determine quality of implementation and architecture

- Automated test coverage
- Number of tests written before coding
- Unit / regression test coverage
- Build time
- Defect density
- Code churn
- Code ownership
- Code complexity
- Coding standards adherence
- Crash rate / time to restore service
- Build breaks / change fail rate
- Technical drag
- Ratio of fixing work vs feature work

Andy Cleff

andycleff@icloud.com
andycleff.com
linkedin.com/in/andycleff
@JustSitThere

12 Rules for Measurement

Rule 1: Measure for a purpose

You must always understand why you are measuring. The metric is not a goal in itself. Never forget that it's just a means to an end. It all starts with why.

Rule 2: Shrink the unknown

A metric is just a surrogate for what you really want to know. Don't jump to conclusions. Always try to reduce the size of what is still unknown.

Rule 3. Seek to improve

Don't only measure things that will make you look good. There is plenty of data around, but you must focus on what enables you to do better work.

Rule 4: Delight all stakeholders

Your work depends on others, and others depend on you. Never optimize for just one stakeholder. Instead, measure your work from multiple perspectives.

Rule 5: Distrust all numbers

Observers usually influence their own metrics, and they suffer from all kinds of biases. Have a healthy, skeptical attitude towards any reported numbers.

Rule 6: Set imprecise targets

When people have targets, they have an inclination to focus on the targets instead of the real purpose. Avoid this tendency by keeping your targets vague.

Rule 7: Own your metrics

Everyone is responsible for their own work, and metrics help us improve that work. Therefore, everyone should be responsible for their own metrics.

Rule 8: Don't connect metrics to rewards

Rewards often kill intrinsic motivation and lead to dysfunctional behaviors in organizations. Don't incentivize people to do work they should like doing.

Rule 9: Promote values and transparency

Human beings are smart and able to game any system. To prevent gaming, be transparent about values, intentions, and the metrics everyone is using.

Rule 10: Visualize and humanize

Numbers tend to dehumanize everything. Replace digits with colors and pictures, and keep the measurements close to where the actual work is done.

Rule 11: Measure early and often

Most people don't measure often enough. Measure sooner and faster to prevent risks and problems from growing too big for you to handle.

Rule 12: Try something else

It's rarely a good idea to do the same things over and over. The environment changes all the time. The same should apply to how and what you measure.

When selecting metrics, ask:

- Why "this metric?" – Why does it matter? Who does it matter to?
- What insights might we gain from it?
- What is expected to change? What is the expected variability and consistency – are we looking for trends or absolute values?
- How might it be gamed, misused (or abused)?
- What are some for trade offs / costs of improvement?
- How often would we like to "take a data point"?
- How long will we run the experiment? (What is the half-life?)
- How when we know when we're "done" with this metric?
- Are we adding to the dashboard or replacing/retiring something else?
- How will we make our measurements transparent – to promote knowledge sharing, collaboration with other teams and trust with our sponsors?
- Is this metric a leading or lagging indicator?

MANAGEMENT 3.0
CHANGE AND INNOVATION PRACTICES