DevOps

Systems Analysis & Design

Learning Objectives

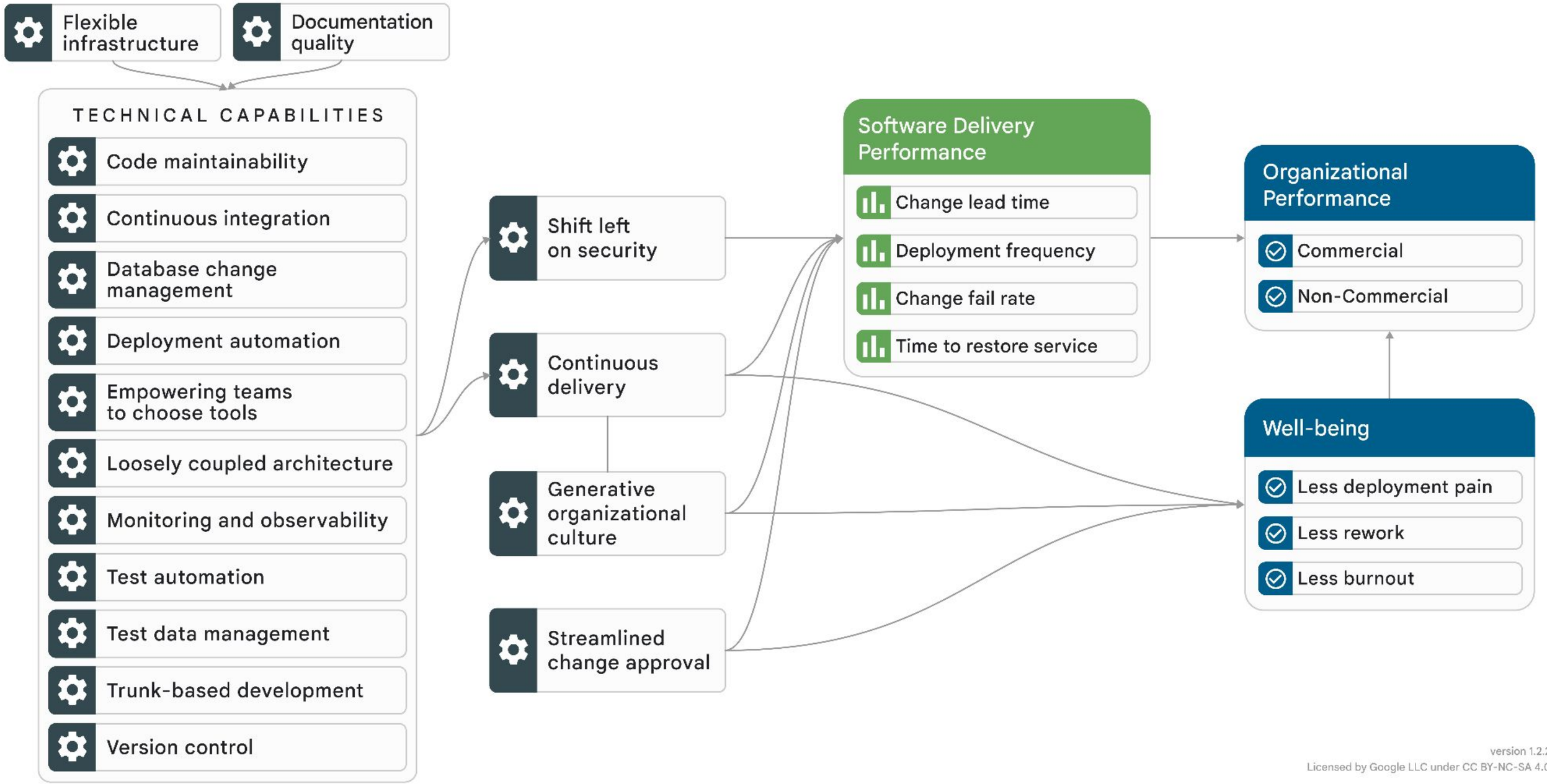
By the end of this session, you will have acquired the following information:

- DevOps
- Software Delivery Performance Metrics
- DevOps Research and Assessment (DORA)

DORA Core Model

dora.dev/research

capabilities → predict → performance → predicts → outcomes



Software Delivery Performance Metrics

- The four metrics of software delivery performance can be considered in terms of **throughput** and **stability**.
- We measure throughput using:
  - Deployment frequency
  - Lead time for changes
- We measure stability using:
  - Time to restore service
  - Change failure rate

Software Delivery Performance Metric

Software Delivery Performance Metric	Low	Medium	High
How often does your organization deploy code to production or release it to end users?	Between once per month and once every 6 months	Between once per week and once per month	On-demand (multiple deploys per day)
What is your lead time for changes (i.e., how long does it take to go from code committed to code successfully running in production)?	Between one month and six months	Between one week and one month	Between one day and one week
How long does it generally take to restore service when a service incident or a defect that impacts users occurs (e.g., unplanned outage or service impairment)?	Between one week and one month	Between one day and one week	Less than one day
What percentage of changes to production or released to users result in degraded service (e.g., lead to service impairment or service outage) and subsequently require remediation (e.g., require a hotfix, rollback, fix forward, patch)?	46%-60%	16%-30%	0%-15%

Documentation Quality

1. I can rely on technical documentation for work.
2. Our technical documentation is easy to understand.
3. Finding the right technical document is easy.
4. Most code for our services or applications is documented.
5. Our technical documentation is well-organized.
6. Documentation is updated with changes.
7. Documentation accurately reflects our primary service or application.
8. I use documentation for troubleshooting incidents or problems.

Flexible Infrastructure

1. I can adjust resources for my service/product based on demand.
2. I can monitor/control resource quantity/cost for my service/product.
3. I can independently provision/configure resources for my service/product on demand.
4. My service/product’s infrastructure serves multiple teams/applications, with dynamic resource assignment.

Trunk-Based Development

1. All developers on my team push code to trunk / main branch at least daily
2. Branches and forks have very short lifetimes (less than a day) before being merged to the main branch
3. There are fewer than three active branches on the application's code repo

Continuous Integration

1. Automated builds and tests are executed successfully every day
2. Automated test failures will block a commit's progress through the pipeline
3. Code commits result in a series of automated tests being run
4. Code commits result in an automated build of the software

Continuous Delivery

1. Fast feedback on the quality and deployability of the system is available to anyone on the team
2. My team prioritizes keeping the software deployable over working on new features
3. Our software is in a deployable state throughout its lifecycle
4. We can deploy our system to production, or to end users, at any time, on demand
5. When people get feedback that the system is not deployable (such as failing builds or tests), they make fixing these issues their highest priority

Generative Organizational Culture

1. Cross-functional collaboration is encouraged and rewarded
2. Failures are treated primarily as opportunities to improve the system
3. Information is actively sought
4. Messengers are not punished when they deliver news of failures or other bad news
5. New ideas are welcomed
6. Responsibilities are shared

Further Resources

- DevOps Research and Assessment (DORA): https://dora.dev/research/