DevOps with Software, Data, and Models

aka DevSecAIOps, DevSecOps
for AI/ML, MLOps

Mathias Kölsch, PhD – 2021

NPS
PRAESTANTIA PER SCIENTIAM
1909

JAIC

What I am about to tell you:

DevOps is crucial for agility

Agility is necessary to achieve readiness and responsiveness.

Data, models, and security must be first-class considerations

Owing to their dissimilarity to software and increased complexity.

Workforce education and close collaboration is essential

Culture shift starts with customer relationships and data entry.

Special thank you to Erin Paciorkowski, JAIC

What is DevOps?

Automation and monitoring at all steps of software system construction and operation:

- Coding & implementation
- Component integration
- Testing and QA
- Deployment
- Infrastructure management, monitoring, scaling

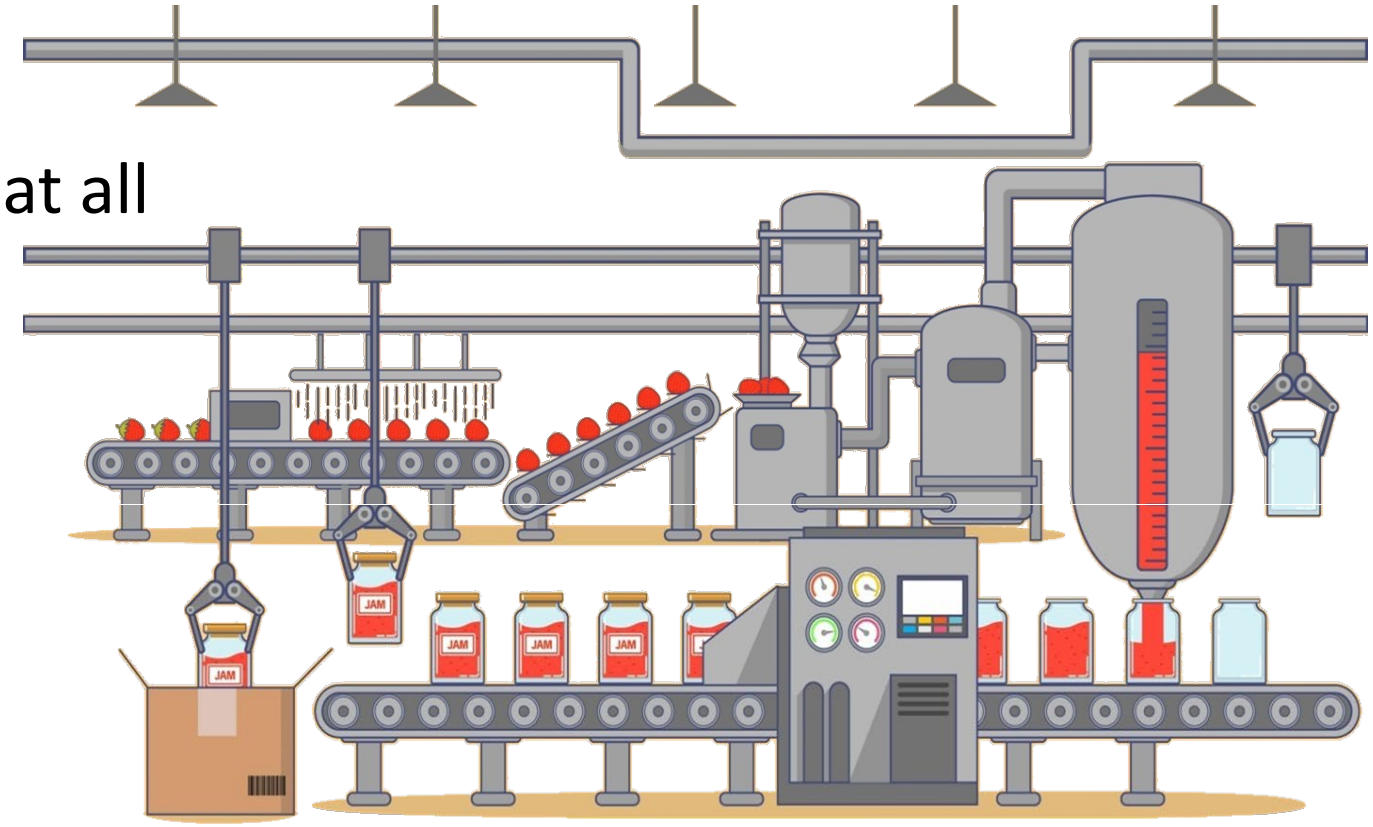
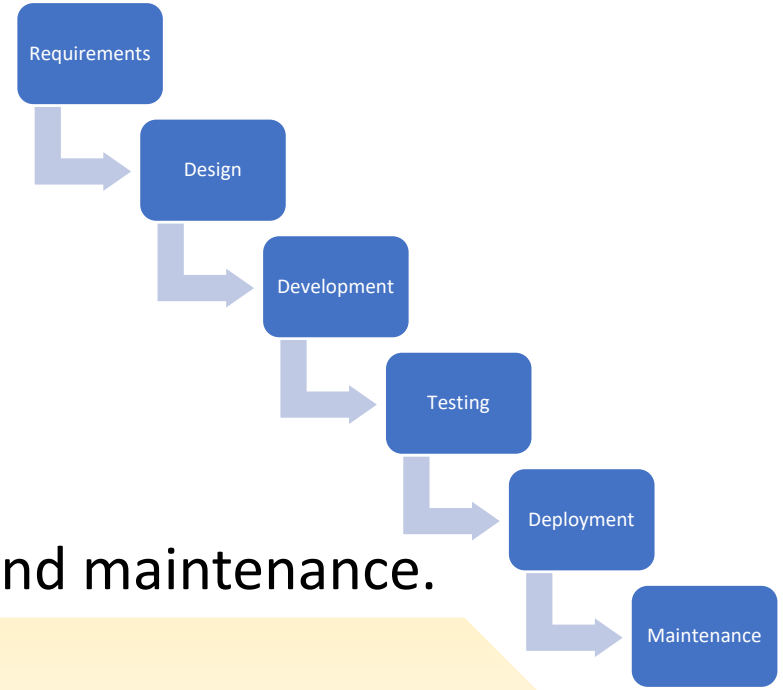


Image Credit: Dukesn, Shutterstock.com

Software Development

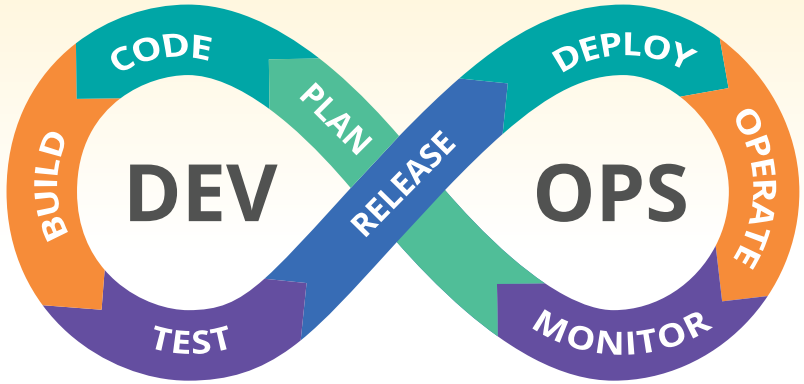


- Traditional SD Life Cycle (SDLC):

  Requirements specification → Implementation → QA

  → Deployment → Usable application. Operation and maintenance.

- DevOps:



Iterate!
Demo the first prototype,
then deliver regular incremental updates.

Planning:
Prioritize features (not releases) regularly for short-term requirements towards long-term goals.

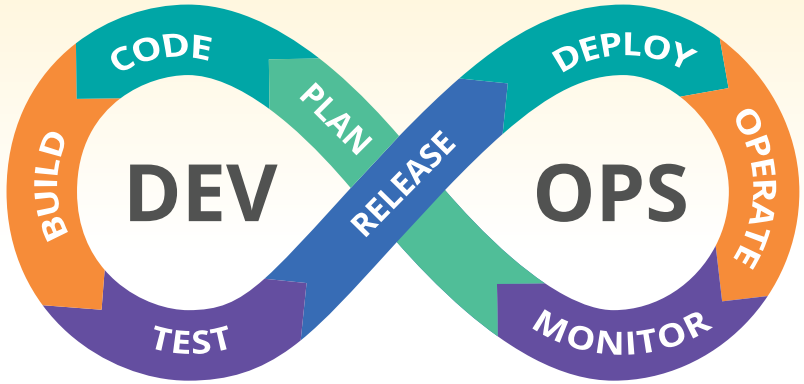
Software Development: Who does what?

- Traditional SD Life Cycle (SDLC):

  Requirements specification → Implementation → QA → Ops → Use

  Customer & product manager → developer → tester → ops team → customer

- DevOps:



Iterate!
Demo the first prototype *to the customer*, then deliver regular incremental updates.

Planning:
Customer & project manager & developer…
prioritize features (not releases) regularly for short-term requirements and long-term goals.

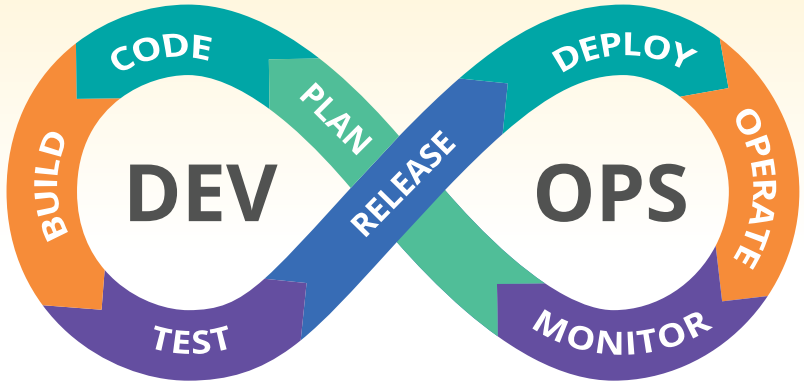
Software Development: Who does what?

- Traditional SD Life Cycle (SDLC):

  Requirements specification → Implementation → QA → Ops → Use

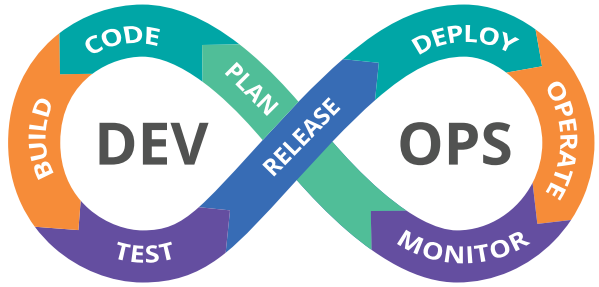
  Customer & product manager → developer → tester → ops team → customer

- DevOps:



Developers write tests and automation
(together with test engineers)
Developers write infrastructure-as-code
(specialized infrastructure engineers)
Developers automate the CI/CD pipeline
(specialized DevOps engineers)
Developers monitor and operate
(together with ops/security team)

DevOps: Why? … and AI/ML



- Avoids late surprises, costly fixes and patches, overrun deadlines
- Avoids requirements creep and feature bloat
- Faster deployment times by leveraging Continuous Integration & Continuous Deployment (CI/CD)
- Use of containers and immutable infrastructure gives high consistency in build and deployment procedures
- Fosters close collaboration of cross-functional multi-org teams
- Enforces and monitors compliance with security, requirements tests, and development standards through scripted steps
- Shift Left: earlier and repeated QA, testing, release, customer validation, etc. since it's part of every loop iteration

DevOps Untruths

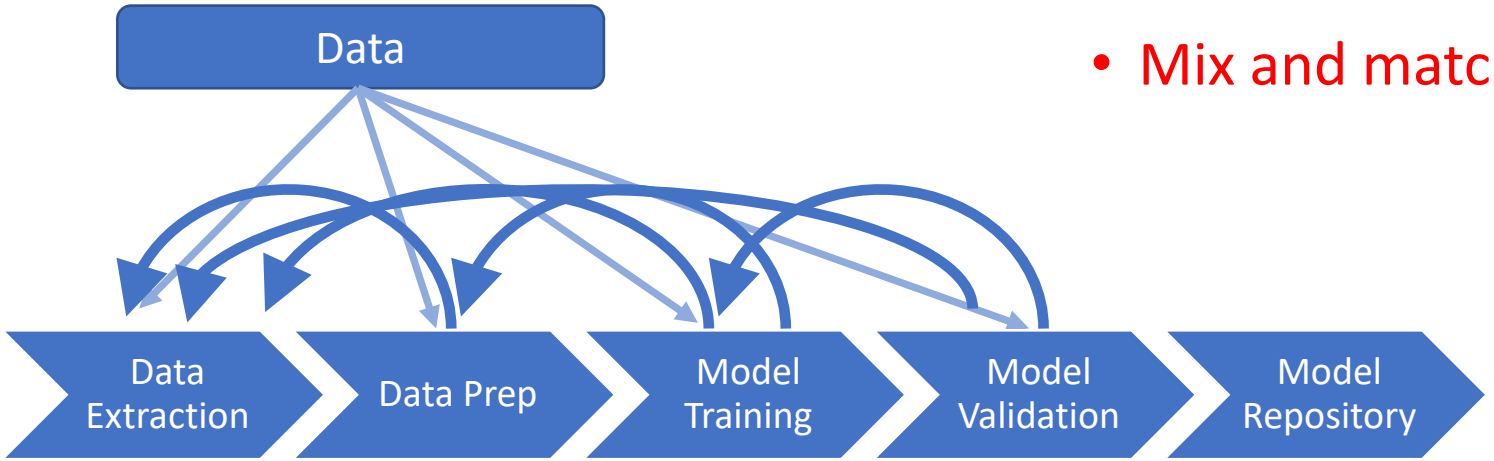
Common DevOps misconceptions:

- ✗ DevOps for AI/ML will solve all your software product development headaches
- ✗ Customers can expect a newly requested feature in the next release
- ✗ Security and testing are built into the pipeline and hence automatic
- ✗ Once the CI/CD pipeline is automated, it will run hands-off
- ✗ Infrastructure as code means that IT administration is a task of the past
- ✗ Cloud-hosted applications are inherently scalable

DevOps and AI/ML

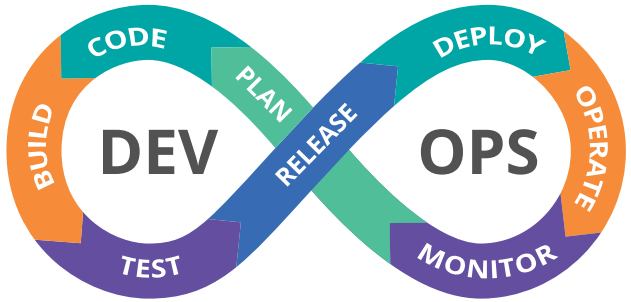
DevOps

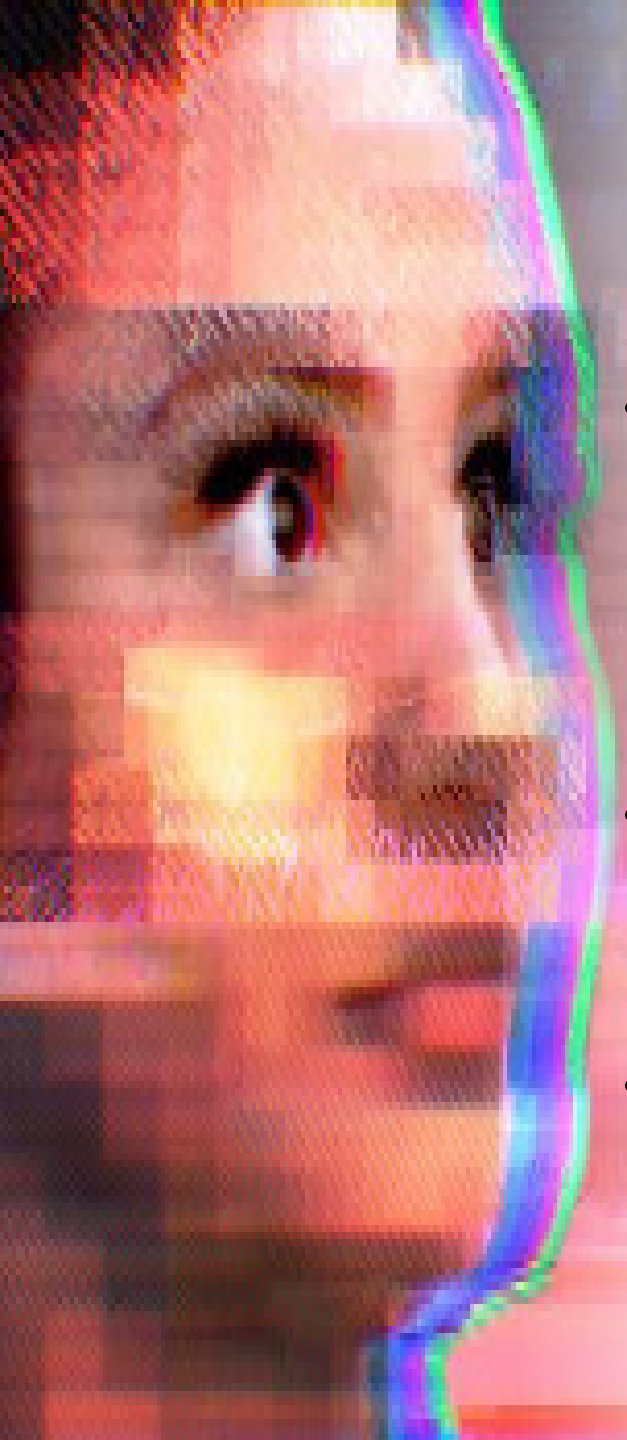
- System behavior depends on source code and configuration
- Pipeline is fixed and one-way



DevOps with AI/ML component

- System behavior depends on source code, configuration, and data
- Manual steps of data exploration and ML experimentation
- Data and Model traceability (pedigree)
- Mix and match data, models, and code





DevOps and AI/ML (continued)

- The manual exploration and experimentation process is essential to AI/ML (yes!) and should not be impeded by a rigid pipeline.
  - Hence, data and models are usually not as well controlled (versioned, updated, reproducible) as is code.
  - "ML is experimental in nature. You should try different features, algorithms, modeling techniques, and parameter configurations to find what works best for the problem as quickly as possible. The challenge is tracking what worked and what didn't, and maintaining reproducibility while maximizing code reusability."
    Google Cloud, MLOps: Continuous delivery and automation pipelines in machine learning
- System testing and validation now needs to take all three into account: code, configuration, and data. Specifically:
  - Data validation (is the data truly what we think it is?)
  - Model evaluation (during and after training)
  - Model validation (while in use)
- Continuous Validation: real-world data characteristics can drift or change suddenly.
  - Model might self-update: "continual learning" with data observed in operation
  - Model behavior needs to be monitored continuously for degradation
  - Example: Microsoft's chatbot Tay

AI and Model Deployment Destinations

- Deploy as Webservice or Micro Service
  - reproducible model deployments with low coupling to client software system
  - highly scalable
- Integrated via Backend Queue
  - inference on backend
  - higher latency than edge
  - efficient scaling
- Deploy to Browser (ex: Tensorflow.js)
  - inference locally
  - no transmission of private data to server
  - transmits model to the end user
- Deploy to Edge (end device)
  - hardware limitations
  - feedback concerns
  - inference on site may be required for some situations
- Deploy to Network Edge
  - 5G provides compute power near end device
  - blend of benefits: low latency, good scalability

Slide modified from Erin Paciorkowski, JAIC

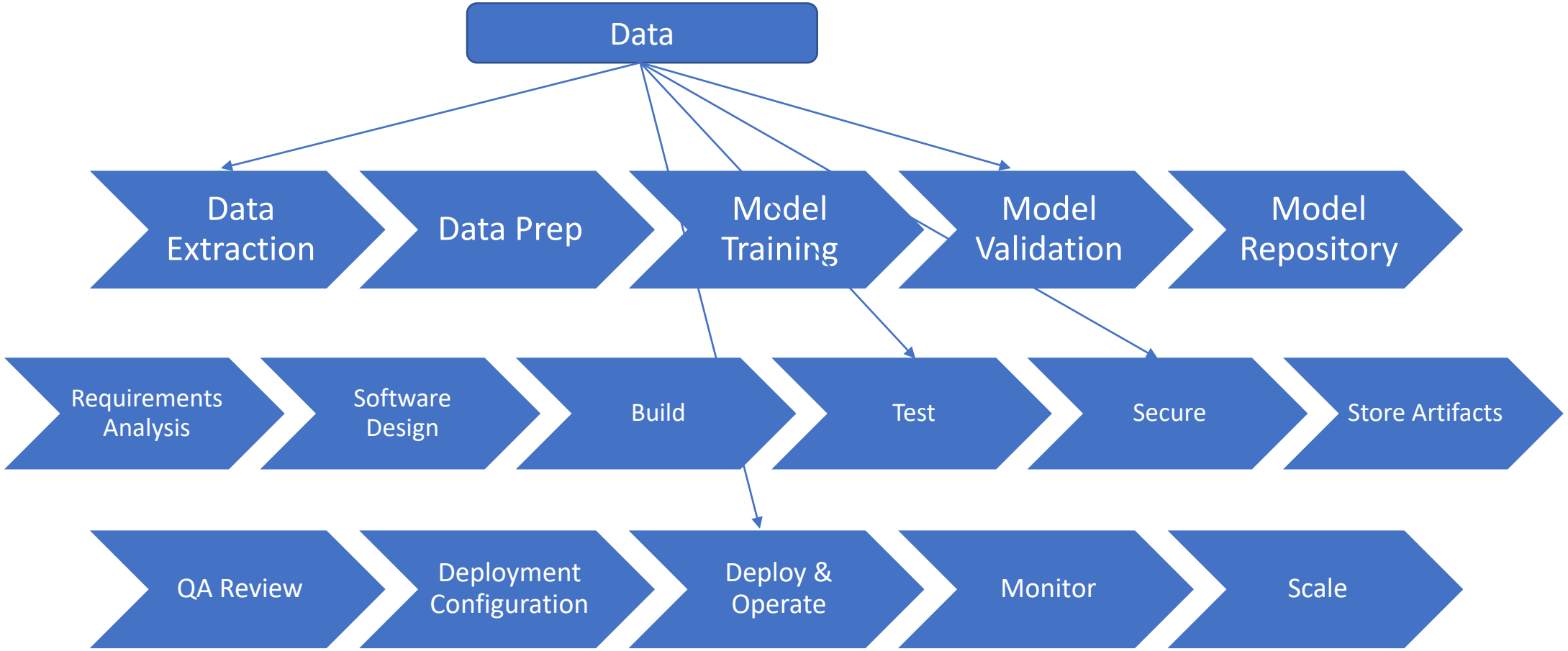
ML Deployment to the Edge

Case Study: 160th Airborne (SOAR), Digital Integrated Maintenance Environment (DIME)

"As part of operationalizing the [ML system], and to provide timely information in an edge environment with limited transport for data backhaul, DIME developers built the Model at the Edge (MaTE) Kit. The MaTE Kit consists of a laptop computer loaded with the [data] download tool, containerized [ML model], and a web-based visualization of AI predictions for operational users. The intent of the MaTE Kit is to provide the user on the ground with an "all in one" tool to interface the aircraft, download data, process and store data, and visualize the [model predictions]."

Jason Slusser: The Power of Data Analytics in Aviation Sustainment. Army Aviation.
http://www.armyaviationmagazine.com/index.php/archive/not-so-current/1849-the-power-of-data-analytics-in-aviation-sustainment

A Long Pipeline



Secure

- Automated testing, vulnerability scans, encapsulation
- Secure: development, data, deployments, operations
- Secret management
- ML specific threat vectors
  - Also covered in other lectures

Pipeline Disambiguation

- DevOps Pipeline
  - Defines the process for software code to turn into a product/service and how to maintain it
  - Mostly synonymous with Continuous Integration (of code and components) and Continuous Delivery (CI/CD)
- MLOps = DevSecAIOps
  - DevOps for systems that have an ML component
- Data Pipeline
  - Data ingest, cleaning, labeling, versioning of curated data sets
- ML training pipeline
  - Upon receiving new (often labeled) data, update or re-train the model
- ML inference (prediction) pipeline
  - Defines data preparation for input into the model, followed by prediction (object detection or regression or ...), and result presentation
- ML model pipeline
  - The part of the MLOps pipeline that creates and tests candidate models, handles model artifacts, configures and deploys the model into the inference pipeline, provides traceability

DevOps, Security, and AI/ML (summary)

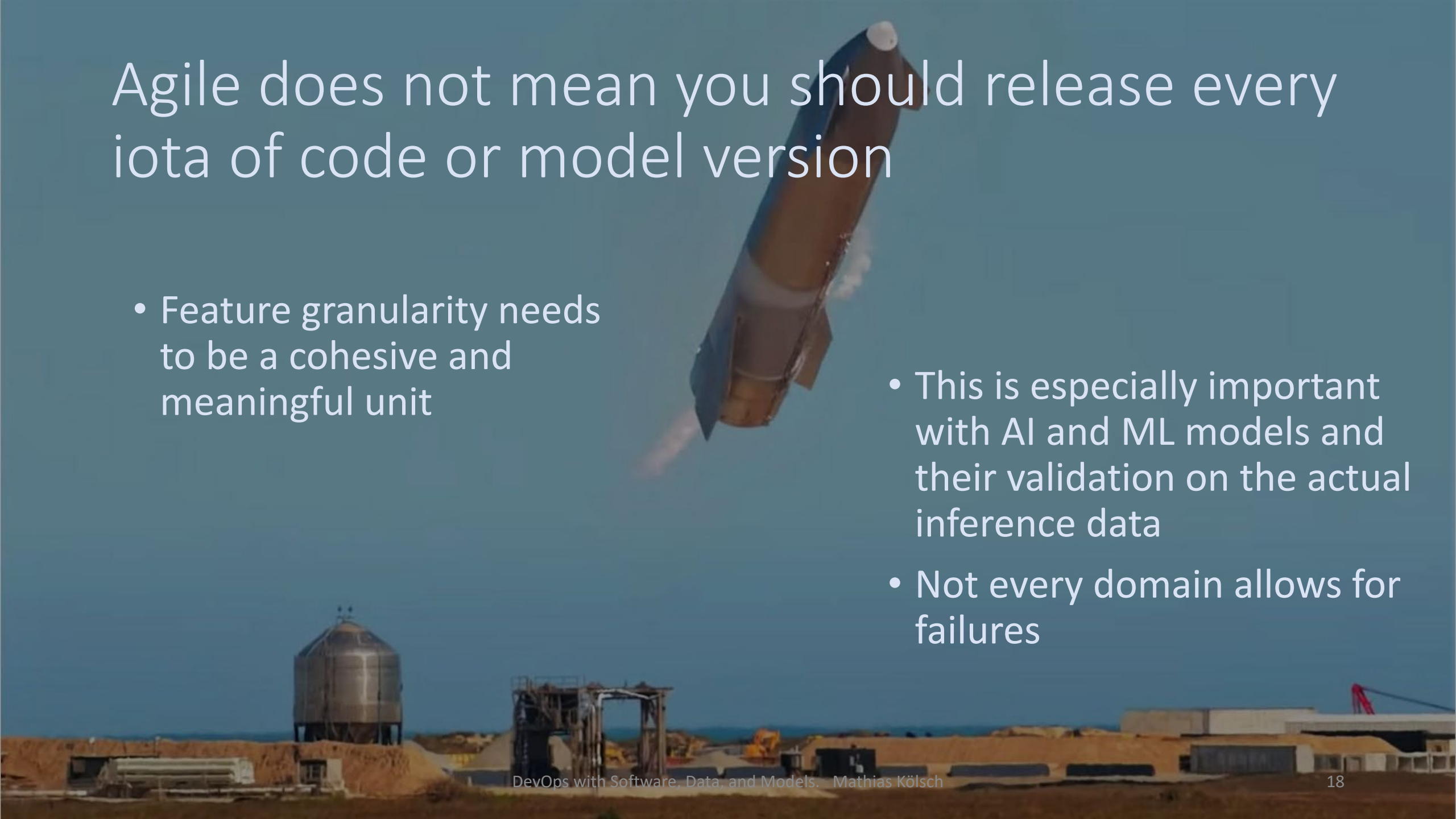
- More components, more steps, more artifacts, more tests, more validation, more configuration
- More people involved (data scientists, ML engineers, model testers, AI ethics reviewer)
- More distinct, asynchronous processes (software dev and data/models) that need to co-evolve
- More frequent updates (of dependencies, data, code, other input)
- More monitoring: “uptime” is insufficient, need to verify real-word results from inference
- More desire to close the loop: observe the performance during inference, and feed newly encountered data back into the pipeline for continual learning
- More demand on traceability and for data pedigree information
- More enforcement of security compliance (continuous checks in the pipeline)
- More expedient vulnerability fixes (to combat zero-day exploits), lower mean-time-to-repair
- More questions, more complexity!

The solution is *not* to give up and to revert to Waterfall.
The solution is to *expect* more DevOps and automation needs,
to use best-practices, proven pipelines, and principles.

Pitfalls

- DevOps should strive for full automation, but that should not be an expected goal
- DevOps should *enable* agile development and CI/CD, but not get in the way of development through…
  - Brittle automation
  - Resource drain on application developers and/or QA testing
- Agile does not remove the need for long-term planning
  - Short-term focus can be distracting from a higher, ultimate goal (compare to company shareholders!)

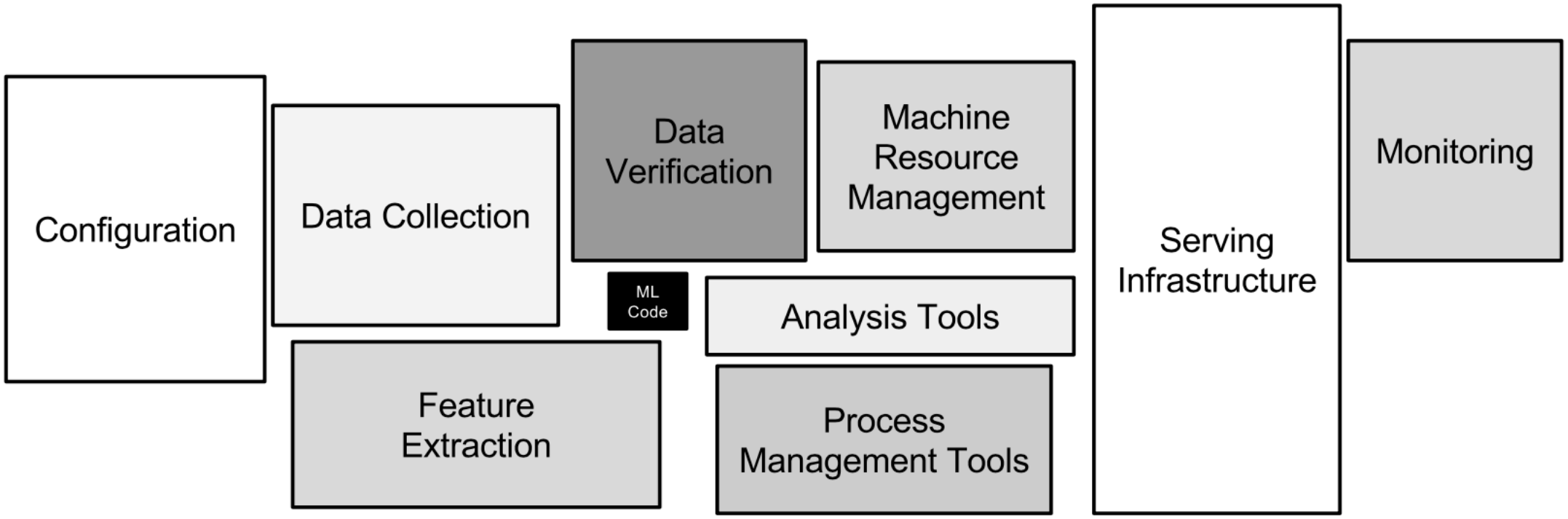
Agile does not mean you should release every iota of code or model version



- Feature granularity needs to be a cohesive and meaningful unit

- This is especially important with AI and ML models and their validation on the actual inference data
- Not every domain allows for failures

Machine Learning in the Big Picture



From Scully et al., Hidden Technical Debt in Machine Learning System, 2015.

Conclusions

- DevOps is crucial for agility
- Security, data and models must be first-class considerations
- Workforce education and close collaboration is essential

- Iterate
- Don’t wait until you know exactly what you want
- Expect more DevOps and automation needs
- Use best-practices, proven pipelines, and available tools

Thank you!

Contact:

Mathias Kölsch, kolsch@nps.edu

Backup Slides

References

- In-depth discussion of MLOps
  - https://papers.nips.cc/paper/5656-hidden-technical-debt-in-machine-learning-systems.pdf
- Data and model versioning:
  - Data Version Control, https://www.dvc.org
  - Mlflow, https://mlflow.org/
- Source control with built-in CI/CD:
  - Gitlab: https://docs.gitlab.com/ee/ci/README.html
  - Github: https://github.com/features/actions
  - Bitbucket: https://bitbucket.org/product/features/pipelines
- CI/CD self-hosted, also available in cloud:
  - Circle CI, https://circleci.com/
  - Jenkins, https://www.jenkins.io/
  - TeamCity, https://www.jetbrains.com/teamcity/
- CI/CD, cloud:
  - https://aws.amazon.com/codepipeline/
  - https://azure.microsoft.com/en-us/services/devops/pipelines/
  - https://cloud.google.com/docs/ci-cd

Self-Assessment Questions

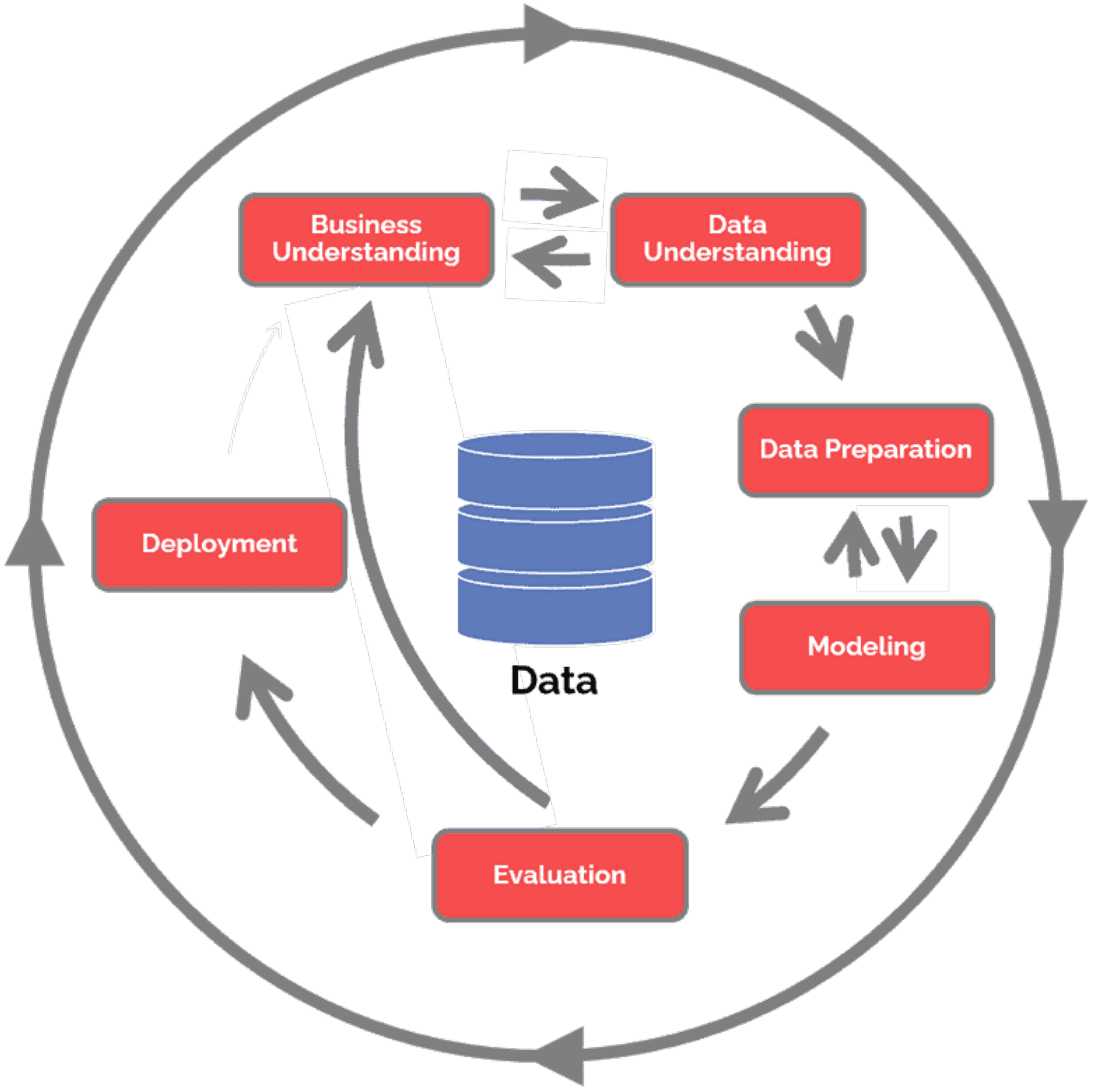
- What takes more person-hours, preparing data or training on data?
- When should the product customer get involved?
  A. Requirements specification
  B. Review of intermediate product prototypes and ML/AI output
  C. At least monthly reviews
  D. No more than quarterly touch points
  E. Final system acceptance review
  F. A, D, E
  G. A, B, C, E
- Software pipeline automation: Check all that apply:
  - Automation is time and resource intensive and is not appropriate for data science/ML/AI projects because they are of more exploratory nature than traditional software products.
  - Full (100%) automation is essential for every step of modern software development, from data ingest to data collection to result delivery.

Data Science Projects in Comparison

	Software Engineering	Data Science
Project Feasibility	Generally known upfront whether a project is executable	Might not be known until late project phases
Focus	Delivering functioning software systems	Delivering actionable insights
Longest Phase	Development (coding)	Data preparation
Scope	Largely defined by stakeholders and product managers	Somewhat define-able by stakeholders and product managers but also needs to be uncovered based on what the data scientists discover
Task Estimation	Task completion time is generally estimate-able	The time required to deliver many steps are unknown
Progress Tracking	Somewhat definitive through metrics like *number of features* or *story points complete*	More ambiguous. Example: Being 50% done with a model doesn't mean anything.
Knowing it works	Mostly binary. Software either works per the specifications or it does not (e.g. the user interface loads or it doesn't)	Many shades of gray. Given a model, one person can say it is working and another could say it is not. Both can be right given their frame of reference.

https://www.datascience-pm.com/data-science-vs-software-engineering/

CRISP-DM



- https://www.datascience-pm.com/crisp-dm-2/