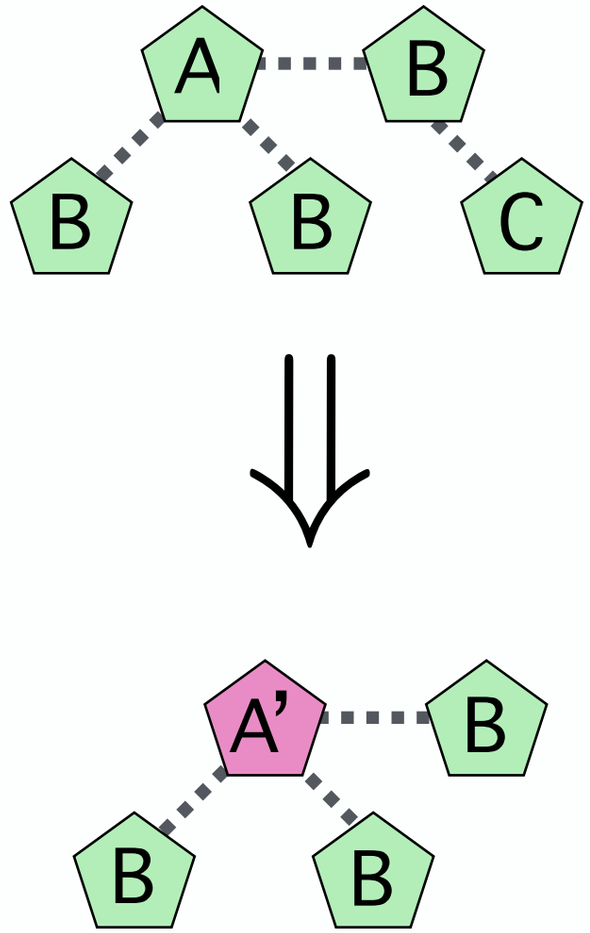


Supporting Microservice Evolution

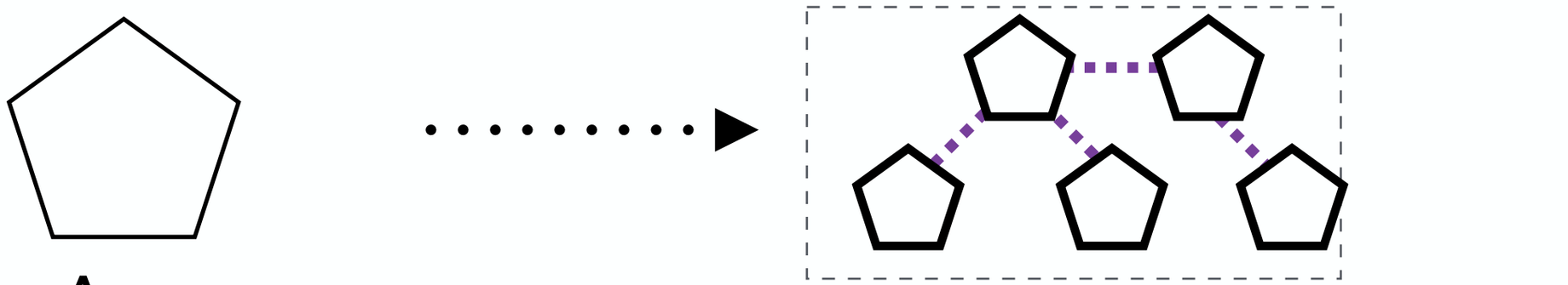
Adalberto R. Sampaio Jr, Harshavardhan Kadiyala, Bo Hu,
John Steinbacher, Tony Erwin,
Nelson Rosa, Ivan Beschastnikh, Julia Rubin

Federal University of Pernambuco Brazil

University of British Columbia Canada

IBM USA/Canada

From monoliths to microservices

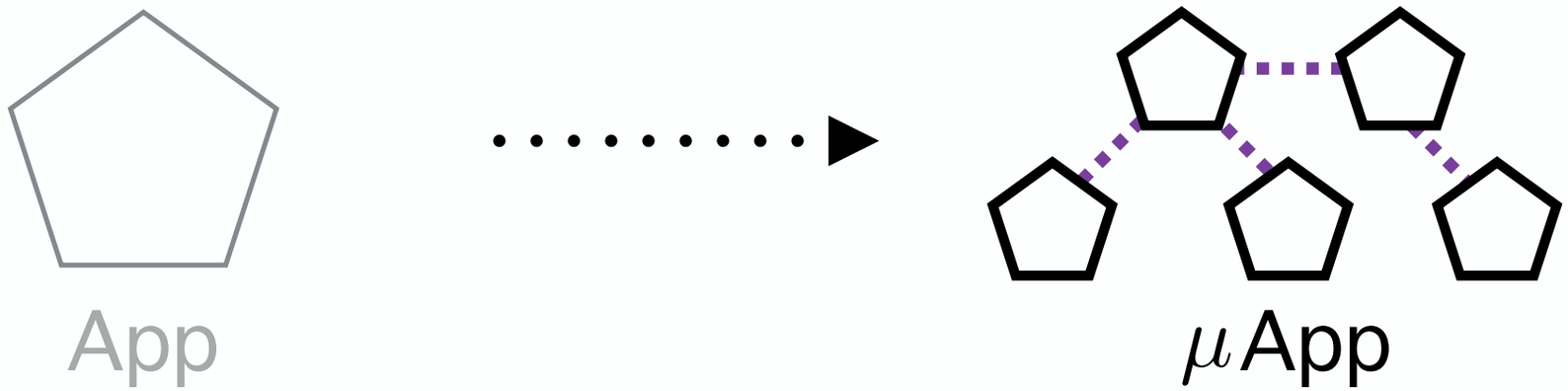


- Microservices (μServices)
  - Fast and easy to deploy
  - Can be scaled independently
  - Multilingual and multi-technology
  - Loose dependencies (REST)

https://martinfowler.com/articles/microservices.html

HUAWEI IBM NETFLIX Hootsuite® ...

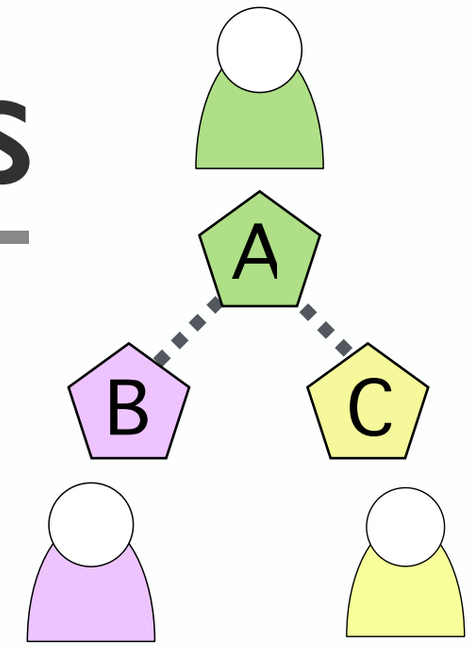
From monoliths to microservices



- Microservices (μServices)
  - Fast and easy to deploy
  - Can be scaled independently
  - Multilingual and multi-technology
  - Loose dependencies (REST)

$\Rightarrow$ Constant evolution

Evolution-related challenges



- Upgrades break inter-services compatibility
  - Different teams work on different services
- Tracking down failure root causes complicated
  - Many moving parts that keep changing
- Deployment configuration must also evolve
  - Changes to performance/dependencies impact config

Existing tool support

- Academia: app-bisect [1], GRU [2], Gremlin [3], Formal methods [4]
- OpenSource (Industry): Istio [5] (IBM/Google), Zipkin [6] (Twitter), Kubernetes [7] (Google), ELK stack [8]

Wanted: program analysis to support change

[1] Rajagopalan and Jamjoom, *App–Bisect: Autonomous Healing for Microservice-Based Aps*, HotCloud 2015

[2] Florio et al., *Gru : an Approach to Introduce Decentralized Autonomic Behavior in Microservices Architectures*, ICAC 2016

[3] Heorhiadi et al., *Gremlin: Systematic Resilience Testing of Microservices*, ICDCS 2016

[4] Panda et al., *Verification in the Age of Microservices,* HotOS 2017

[5] https://istio.io/

[6] http://zipkin.io/

[7] https://kubernetes.io/

[8] https://logz.io/learn/complete-guide-elk-stack/

Existing tool support

- Academia: app-bisect [1], GRU [2]
- OpenSource (Industry): Istio [3] (IBM/Google), Zipkin [4] (Twitter), Kubernetes [5] (Google), ELK stack [6], Zuul [7] (Netflix), InfluxDB [8]

Idea: model μApp over time

Wanted: program analysis to support change

[1] Rajagopalan and Jamjoom, *App–Bisect: Autonomous Healing for Microservice-Based Aps*, HotCloud 2015

[2] Florio et al., *Gru : an Approach to Introduce Decentralized Autonomic Behavior in Microservices Architectures*, ICAC 2016

[3] https://istio.io/

[4] http://zipkin.io/

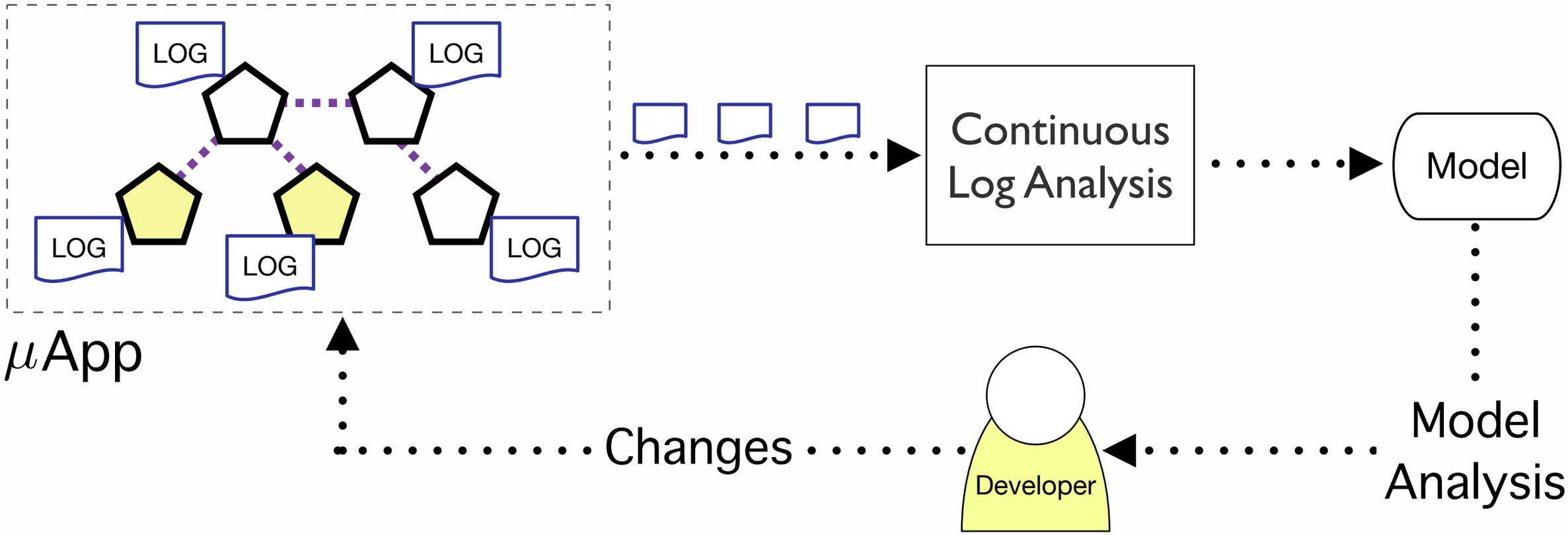
[5] https://kubernetes.io/

[6] https://logz.io/learn/complete-guide-elk-stack/

[7] https://github.com/Netflix/zuul/

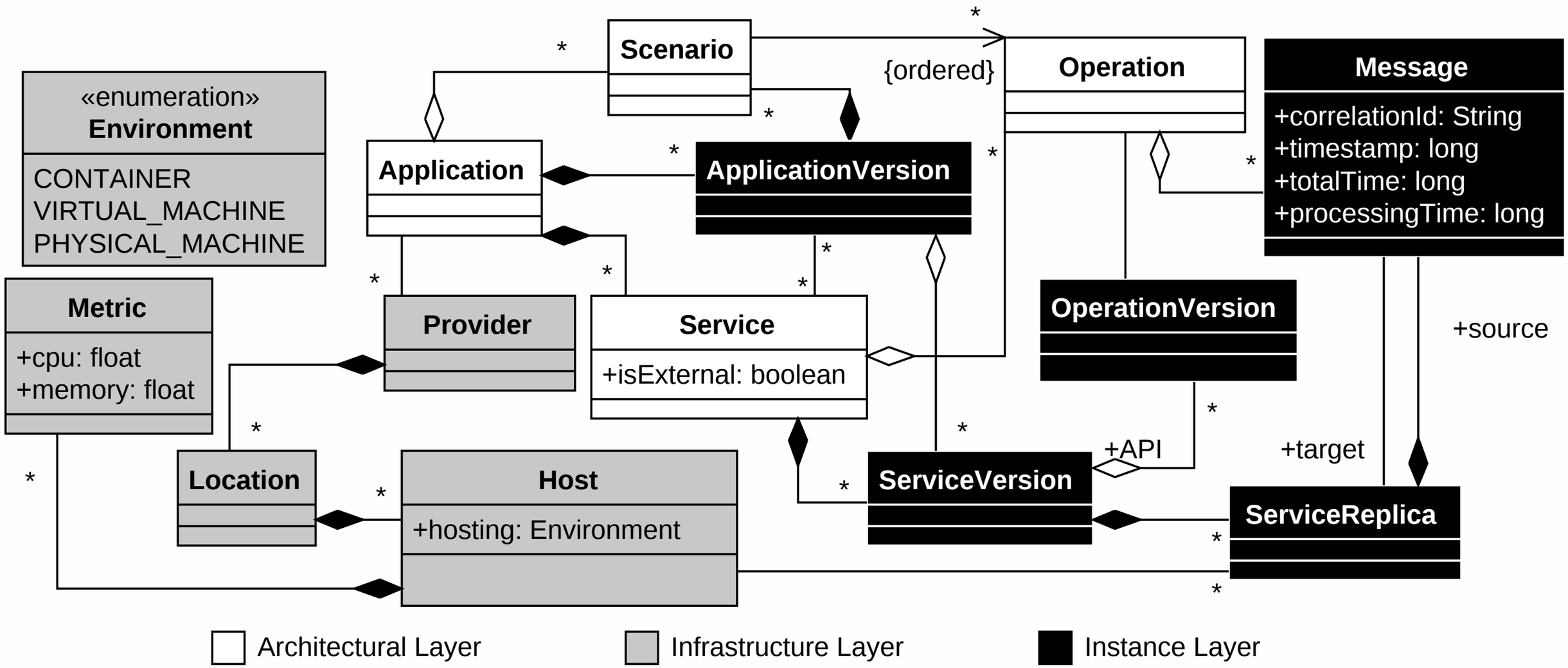
[8] https://www.influxdata.com/

Model μApp over time

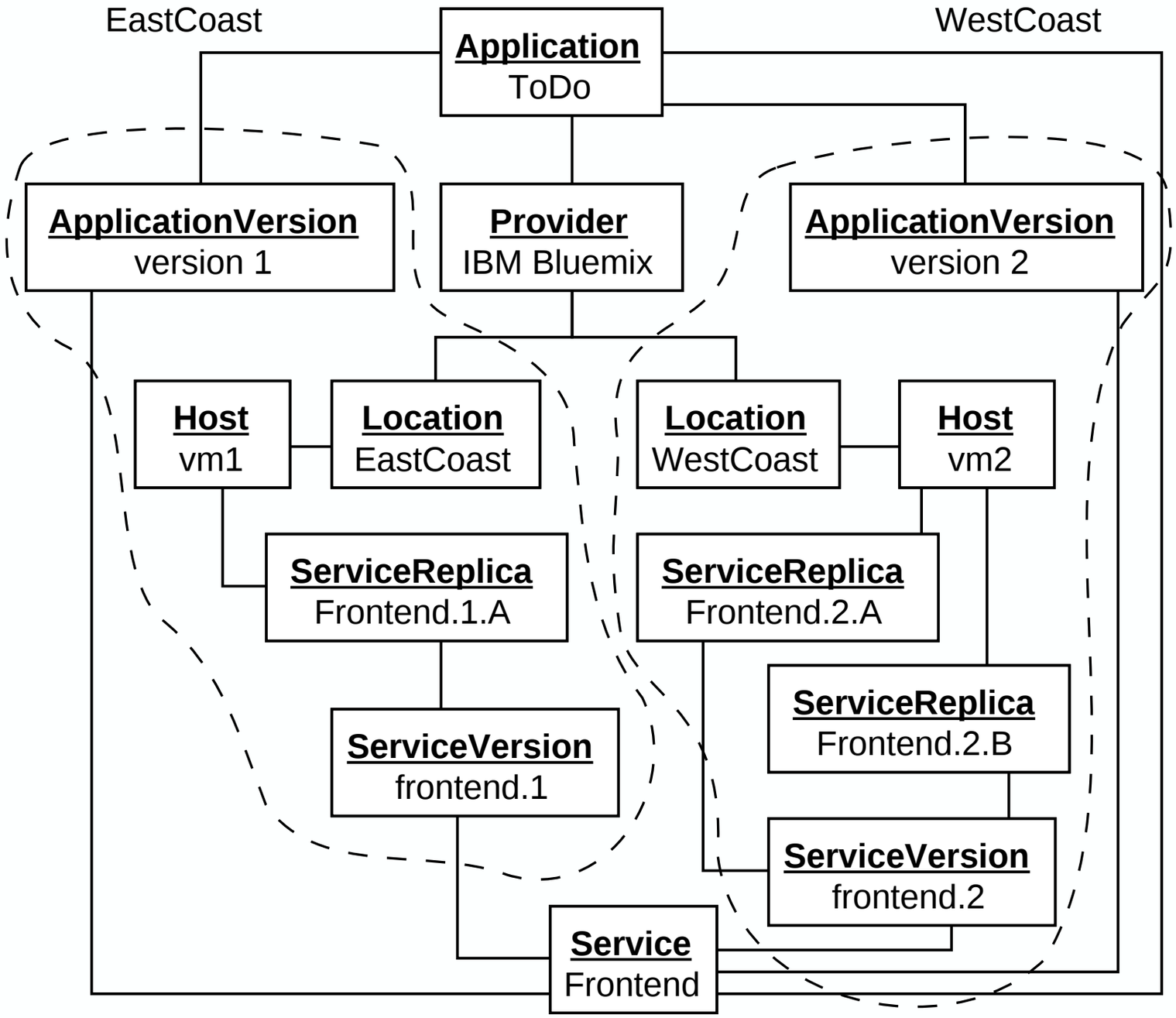


1. Construct a model of the μApp and its configuration
2. Update model as μApp or configuration change
3. Analyze sequence of models (past, present, future)
4. Use the analysis to support developers

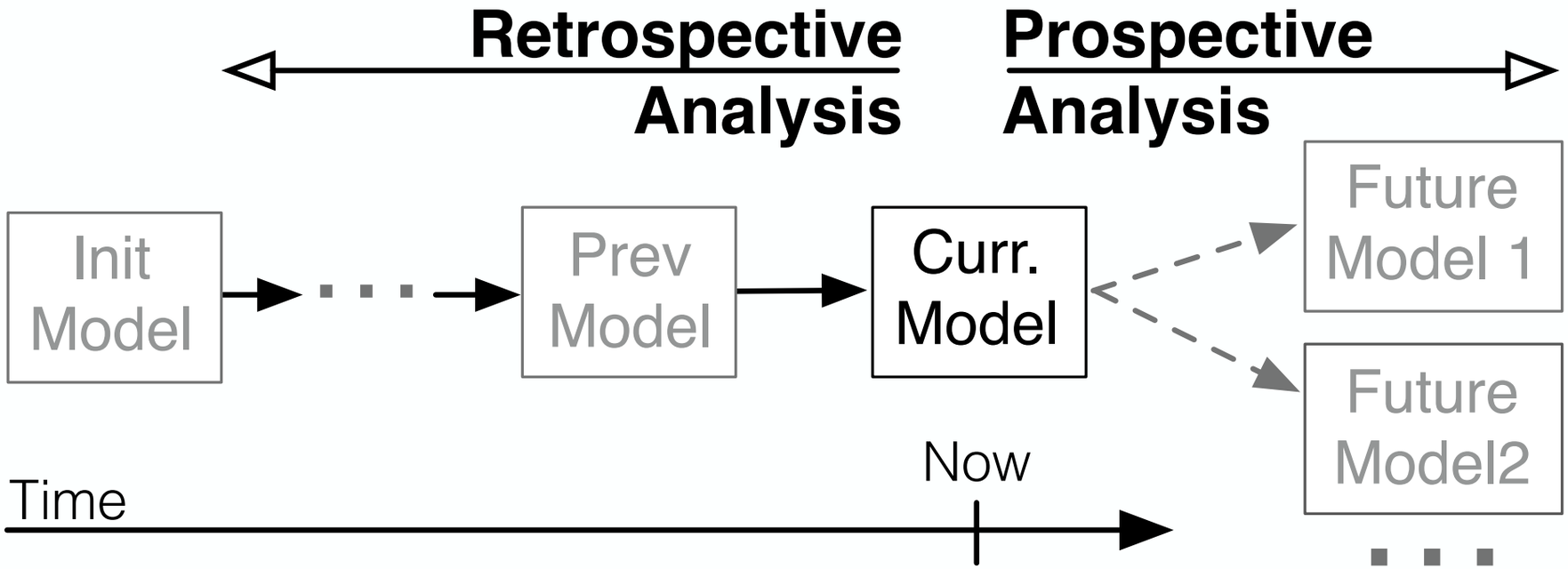
Proposed evolutionary (meta) model



Example model instance



Model analysis



- Sequence of models over time allows for rich analyses
  - Retrospective: Study inter-services messages to recommend service refactorings
  - Prospective: Explore and instantiate new deployment configurations to optimize resources usage

Ongoing/future work

- Assessing developer needs: which tasks are the most pressing?
- Model representation: many modeling formalisms, which one is the best for the task?
- Defining analysis: Build on existing model analysis work
- Extensibility: Can we allow the model to change?
  (Cannot foresee future information we may need to integrate)
- Social factors: interplay between technical and social dependencies

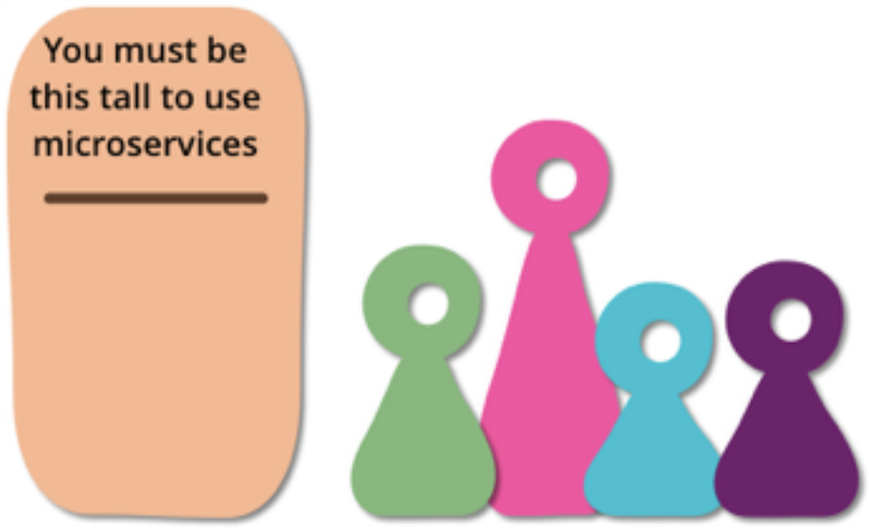
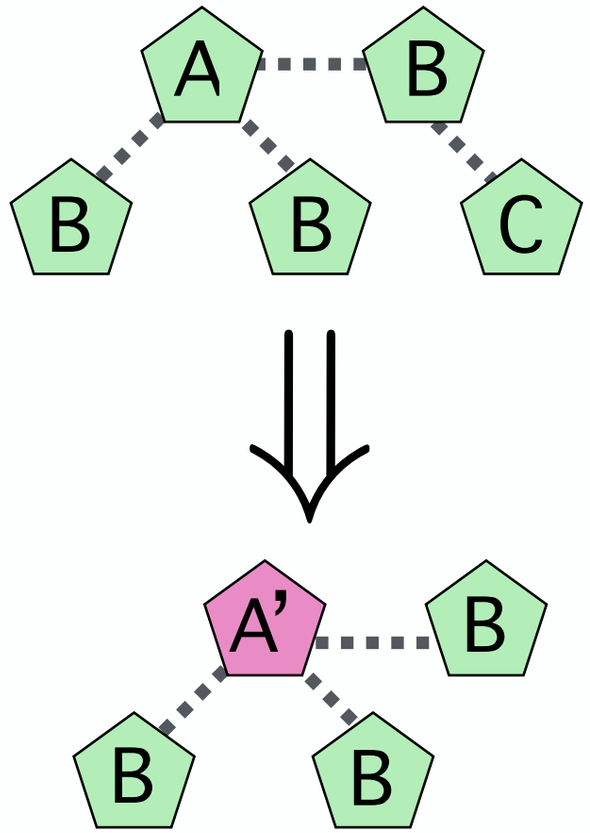


Image from: https://martinfowler.com/bliki/MicroservicePrerequisites.html

Microservices require new approaches

- Loose coupling
- Rapid deployment
- Multi-lingual
- Multiple teams



Constant
Change

Vision: Generate an **evolutionary model** from dynamic observations (logs) for retrospective and prospective analyses

Find us at tomorrow's poster session!