Aalto University
School of Science

Robustness, Reliability, Resilience and Elasticity for Multi-continuum Service-based Applications/Systems

Hong-Linh Truong
linh.truong@aalto.fi

10.09.2025

Learning objectives

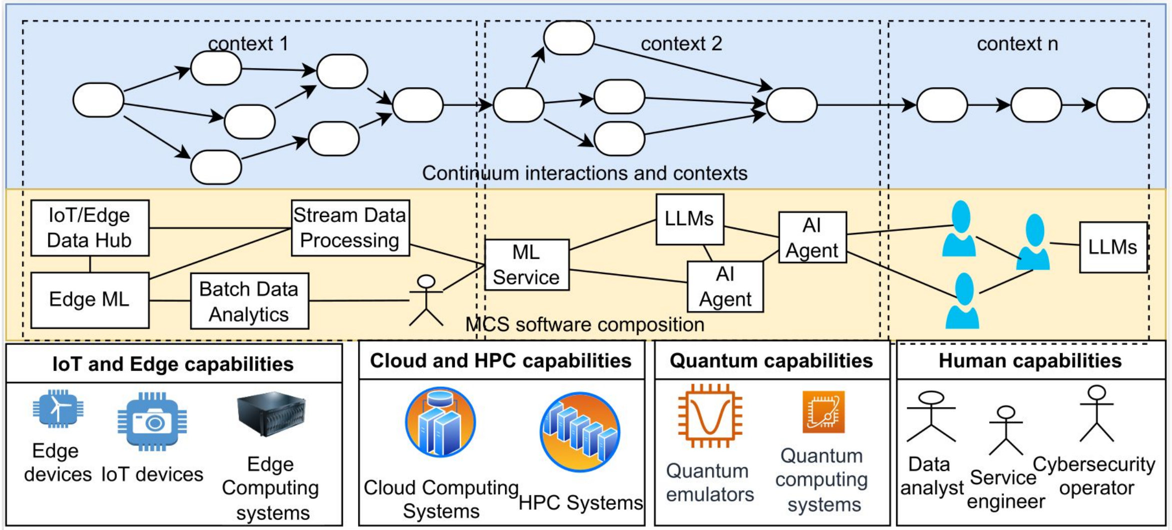
- Understand concepts of robustness, reliability, resilience and elasticity (R3E)
- Understand key steps in engineering R3E
- Understand and able to design and apply coordination techniques for R3E

Content

- Robustness, Reliability, Resilience and Elasticity (R3E)
- R3E engineering for multi-continuum computing
- Coordination for R3E
  - Basic models
  - Some patterns for AI/ML services

Robustness, Reliability, Resilience and Elasticity (R3E)

Multi-continuum computing



Multi-continuum Service-based Applications/Systems

Complex characteristics! so how to build them?

- *Robustness, Reliability, Resilience, & Elasticity (R3E) ⇒ today*
- *Monitoring, Observability, Vulnerability and Explainability*
- *Robustness as a key factor for trustworthiness in hybrid intelligence software*

Objectives for end-to-end systems engineering in multi-continuum computing

- Deal with end-to-end aspects required by the real world
  - e.g., not just software services or AI/ML models and their optimization
- Scale the system
  - computing capabilities in large-scale infrastructures for complex requirements
  - intelligence capabilities
- Optimize the system under various constraints
  - time, cost, accuracy, etc.(quality of analytics) and trade-offs
- Offer a production-level “trustworthy service”

R3E

- Robustness
    - ability to cope with errors
- Reliability
    - ability to function according to the indented specification (in a proper way)
- Resilience
    - “ability to provide the required capability in the face of adversity”(https://www.sebokwiki.org/wiki/System_Resilience)
- Elasticity
    - ability to stretch and return to normal forms (under external forces)

Short summary in big data/ML view

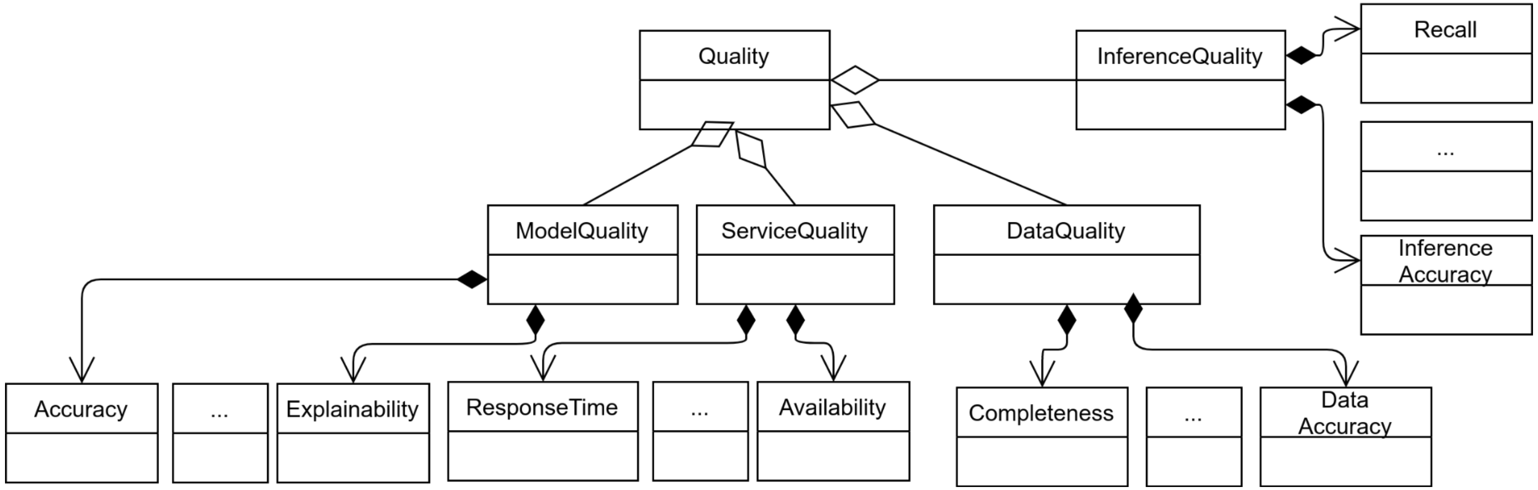
R3E attributes	Cases from big data view	Cases from machine learning view
Robustness	deal with erroneous and bad data (Zhang et al., 2017), data processing job robustness	dealing with imbalanced data, learning in an open-world (out of distribution) situations (Kulkarni et al., 2020; Sehwag et al., 2019; Saria and Subbaswamy, 2019; Hendrycks and Dietterich, 2019)
Reliability	reliable data sources, support of quality of data (Zhang et al., 2020; Lee, 2019), reliable data services (Kleppmann, 2016), reliable data processing workflows/tasks (Zheng et al., 2017)	reliable learning and reliable inference in terms of accuracy and reproducibility of ML models (Saria and Subbaswamy, 2019; Henderson et al., 2017); uncertainties/confidence in inferences; reliable ML service serving
Resilience	software bugs, infrastructural resource failures, fault-tolerance and replication for data services and processing (Yang et al., 2017)	bias in data, adversary attacks in ML (Katzir and Elovici, 2018), resilience learning (Fischer et al., 2018), computational Byzantine failures (Blanchard et al., 2017)
Elasticity	utilizing different data resources; increasing and decreasing data usage with respect to data volume, velocity and quality; elasticity of underling resources for data processing (Wang and Balazinska, 2017)	elasticity of resources for computing (Huang et al., 2015; Harlap et al., 2017; Gujarati et al., 2017), elasticity of model parameters; performance loss versus model accuracy; elastic model services for performance

Source: https://www.researchgate.net/publication/341762862_R3E_-An_Approach_to_Robustness_Reliability_Resilience_and_Elasticity_Engineering_for_End-to-End_Machine_Learning_Systems

Comprehensive quality attributes related to R3E in multi-continuum

Attributes **systematically modeled, programmed & captured** at different levels of abstractions

combined with others to create **quality of analytics (QoA)** for an application/system



QoA: constraints of quality attributes based on requirements that need to be assured

Source: ***"Coordination-aware assurance for end-to-end machine learning systems: the R3E approach"***, AI Assurance, https://doi.org/10.1016/B978-0-32-391919-7.00024-X
"QoA4ML - A Framework for Supporting Contracts in Machine Learning Services", ICWS *2021* doi: 10.1109/ICWS53863.2021.00066.
QoA4ML specs and prototype: https://github.com/rdsea/QoA4ML

New interpretation of R3E for multi-continuum service-based applications/systems

- R3E must cover data, AI/ML models, software services, etc. and their dependencies
  - beyond typical computing resources at the system level
- R3E as key factors of trustworthiness
  - R3E as a means to mitigate potential risks for trustworthiness

R3E as core factors of trustworthiness in AI/ML services

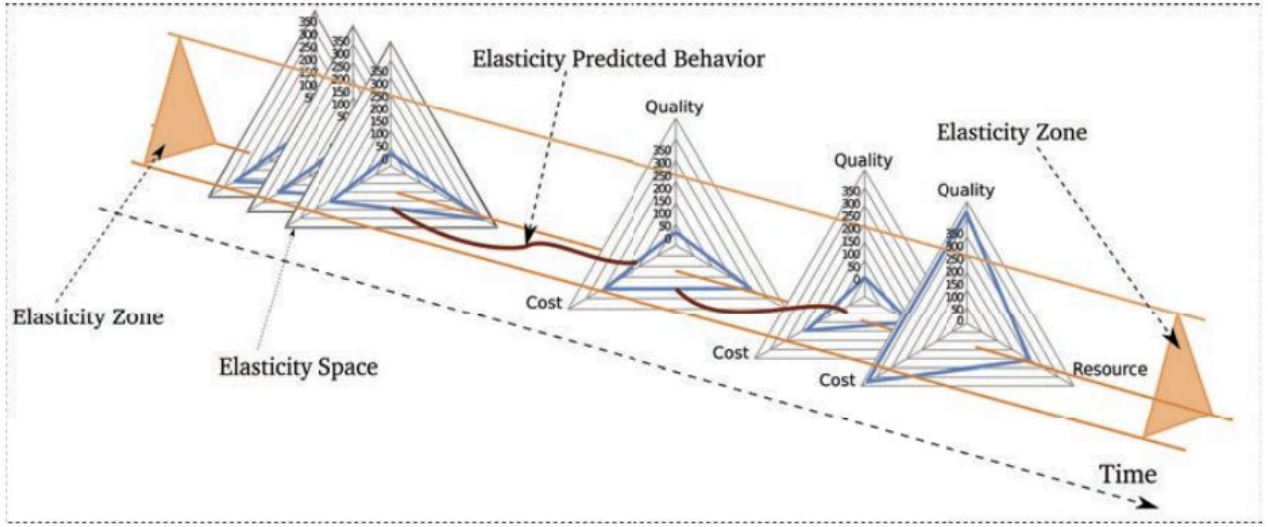
Figure source: https://airc.nist.gov/airmf-resources/airmf/3-sec-characteristics/

R3E engineering

Elasticity

- Elasticity in cloud computing
  - mostly about adding, changing, and removing computing resources to adapt to workloads
    - should be done automatically via rules/algorithms
- Multi-dimensional elasticity in edge-cloud computing
  - resources (computing, data), quality, and cost
- Multi-dimensional elasticity for multi-continuum computing
  - resources offering capabilities: computing, data, and intelligence
  - complex quality and new attributes related to AI/ML
    - quality of analytics (QoA) in general and QoA for AI/ML

Multi-dimensional elasticity in edge-cloud computing



- Multiple attributes in resource, quality and cost dimensions

- Concepts of elasticity space and zone can be extended for multi-continuum computing

Figure source: Truong et al., *"Towards the Realization of Multi-dimensional Elasticity for Distributed Cloud Systems",* https://doi.org/10.1016/j.procs.2016.08.276.

Resilience

Resilience techniques in cloud computing (mainly for computing resources)

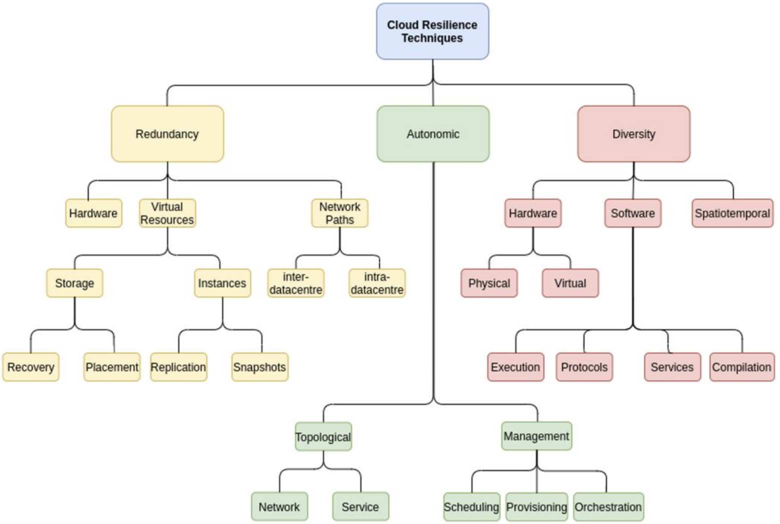


Figure source: Thomas Welsh and Elhadj Benkhelifa. 2020. On Resilience in Cloud Computing: A Survey of Techniques across the Cloud Domain. ACM Comput. Surv. 53, 3, Article 59 (May 2021), 36 pages. https://doi.org/10.1145/3388922

Resilience in AI/ML: security-related adversary

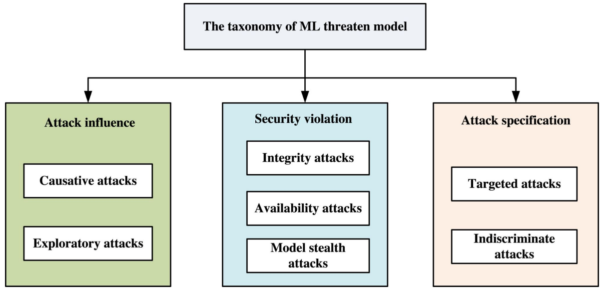
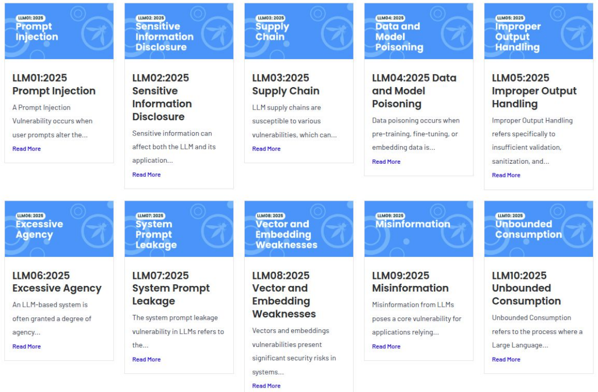


Figure source: Xianmin Wang, Jing Li, Xiaohui Kuang, Yu-an Tan, Jin Li, "*The security of machine learning in an adversarial setting: A survey, Journal of Parallel and Distributed Computing*", https://doi.org/10.1016/j.jpdc.2019.03.003.



Source: https://genai.owasp.org/llm-top-10/

Top 10 Machine Learning Security Risks

- ML01:2023 Input Manipulation Attack
- ML02:2023 Data Poisoning Attack
- ML03:2023 Model Inversion Attack
- ML04:2023 Membership Inference Attack
- ML05:2023 Model Theft
- ML06:2023 AI Supply Chain Attacks
- ML07:2023 Transfer Learning Attack
- ML08:2023 Model Skewing
- ML09:2023 Output Integrity Attack
- ML10:2023 Model Poisoning

Source: https://owasp.org/www-project-machine-learning-security-top-10/

R3E engineering (1)

Designing and programming R3E components

Automatic deployment and configuration

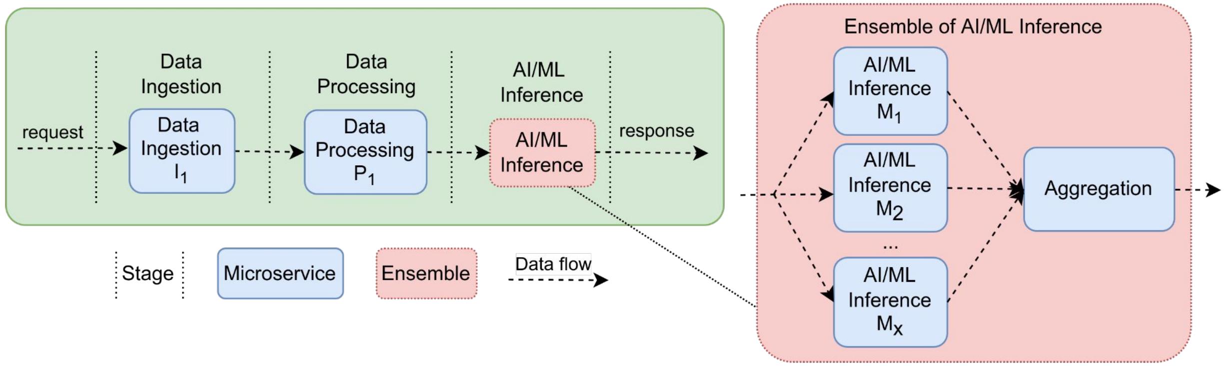
Monitoring and analysis

Coordinated control for R3E

R3E engineering (2)

- Identifying, conceptualizing, and modeling R3E objects
  - AI/ML models, computing resources, data and QoA metrics
- Defining and capturing R3E primitive operations
  - change capabilities, QoA metrics, AI/ML model parameters, input data
- Programming features for R3E objects
  - with control/data flows, coordinating quality of analytics (QoA) adjustment, dynamic service serving models
- Runtime deploying, control, and monitoring techniques for R3E objects

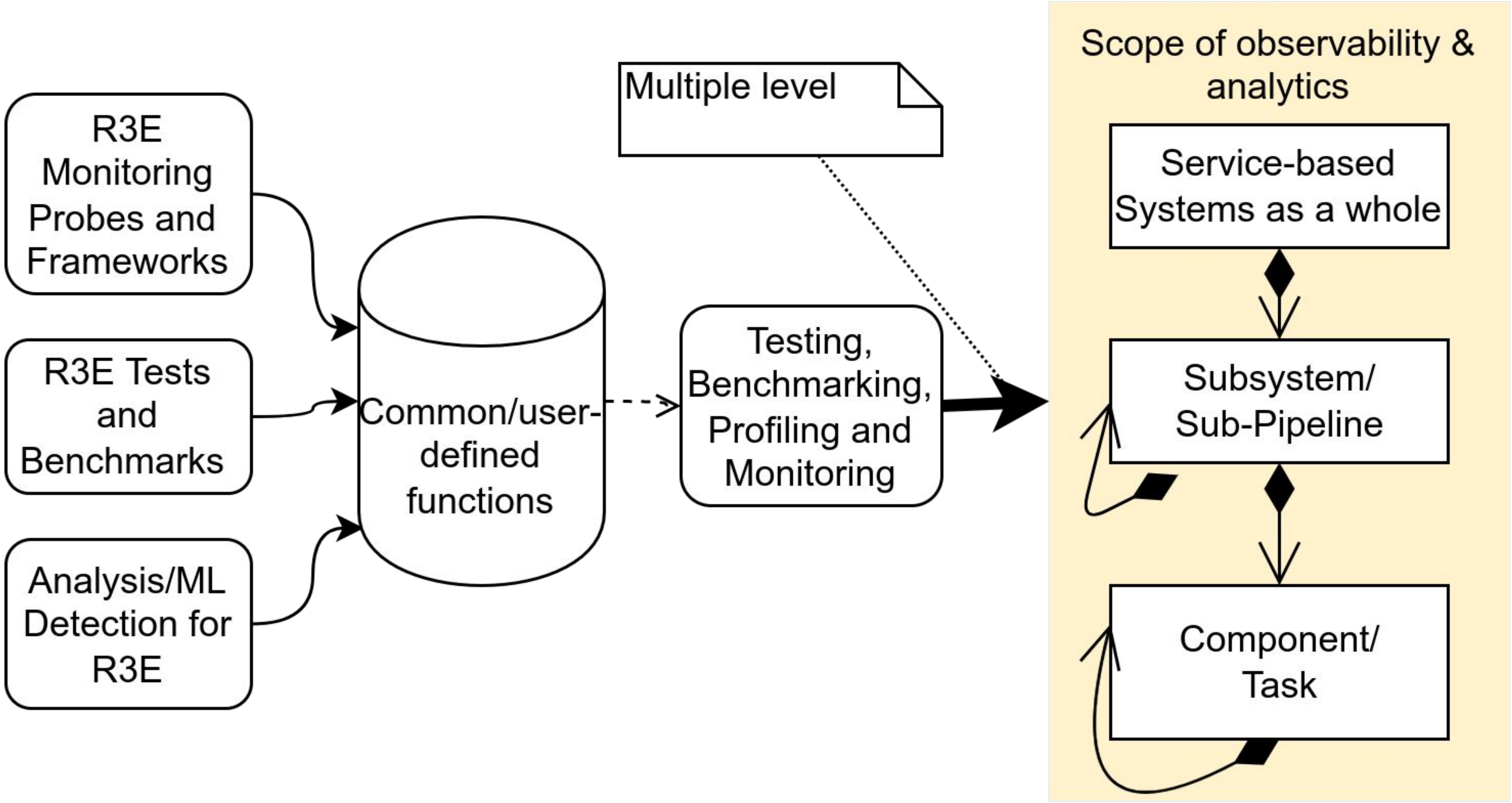
Example: AI/ML object detection



Goal: dealing with elasticity but trade-off for cost, accuracy and time

- R3E objects for data ingestion
  - data is stored into files or messaging
- R3E objects for data processing
  - simple, multi-processes, workflow based data processing
- R3E objects for AI/ML inference
  - Yolo-based object detections
    - each version has different abilities w.r.t. which classes of objects to be detected
    - complementary vs competitive capabilities from the detection viewpoint
- all may require different computing resources (and consequently cost and performance differences)

Multi-level monitoring and analysis for R3E



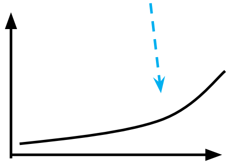
Example: detecting/profiling elasticity

Elasticity Space Func (from when to when?)

Change point detection algs

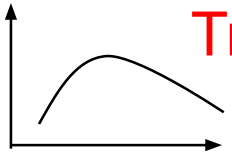
Elasticity Trend Func

cost

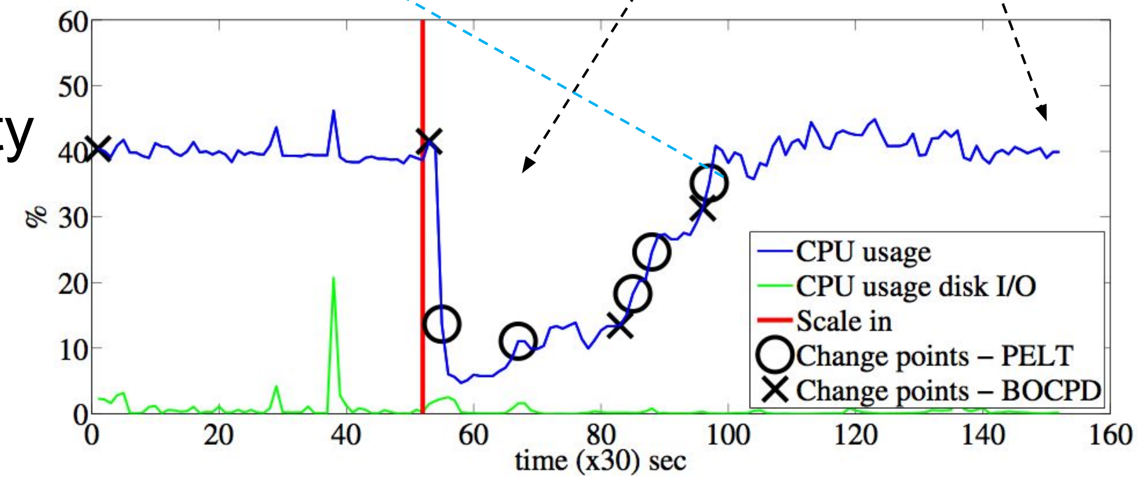


Trend 1 quality

cost

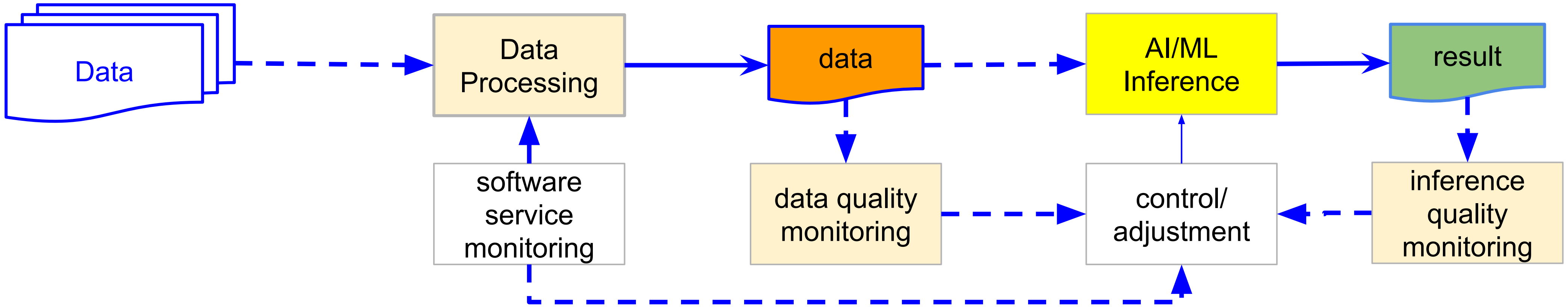


Trend 2 quality



Alessio Gambi et al.: "*On estimating actuation delays in elastic computing systems*". SEAMS 2013: 33-42

Example: detecting quality of intermediate/final data or inference results, dealing with robustness/reliability in data



- need different tools/techniques to evaluate quality of data
  - profiling, sampling, drift detection
  - very tricky with streaming data: quality vs overhead
  - very different from software performance evaluation

Example: policies for monitoring R3E attributes

Changed at runtime and easily be combined with *other types of policies in Infrastructure-as-Code* in edge-cloud continuum

```json
"resources": {
   "mlmodels":[
     {"id": "ml_inference", "mlinfrastructures": "tensorflow", "machinetypes":
     ["edge", "cloud"], "inferencemodes": "dynamic"}
   ]
 },
"quality": {
   "services": [
          {"ResponseTime":{"operators":"max","unit":"s","value":0.05,"class":
          ["performance"], "machinetypes": ["edge"]}}
   ],
   "data":[
          {"Accuracy":{"operators":"min","unit":"percentage","value":80,
          "class": ["qualityofdata"]}}
   ],
   "mlmodels": [
          {"Accuracy":{"operators":"min","unit":"percentage","value":80,
          "class": ["Accuracy"], "machinetypes": ["edge"]}}
   ]
}
```

Rego based implementation

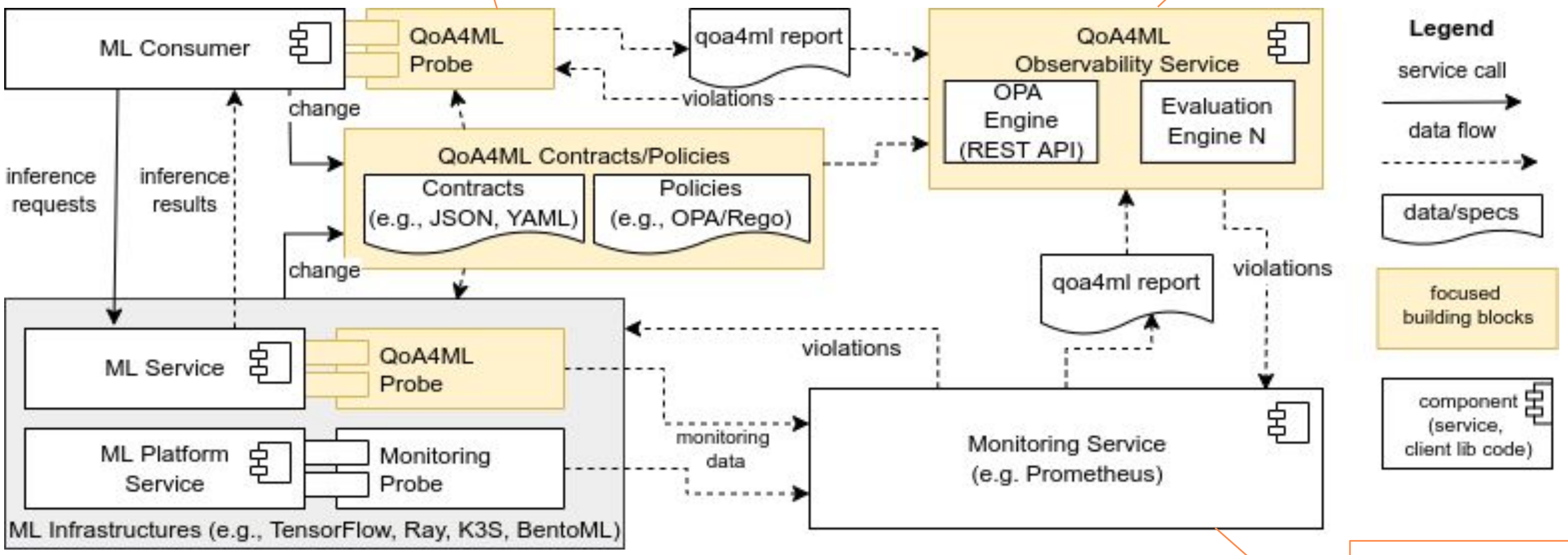
```rego
package qoa4ml.bts.alarm
import data.bts.contract as contract
default contract_violation = false
contract_violation = true {
    count(violation) > 0
}
# The policy checker will receive the contract and the runtime information
# input variable: the input of runtime metrics
violation[[input.client_info, "Accuracy violation on edge resource"]]{
    input.service_info.machinetypes == "edge"
    input.service_info.metric[_] == "Accuracy"
    some i, j
    contract.quality.mlmodels[i].Accuracy.machinetypes[j] == "edge"
    input.metric.Accuracy < contract.quality.mlmodels[i].Accuracy.value
}
violation[[input.client_info, "ResponseTime violation on edge resource"]]{
    input.service_info.machinetypes == "edge"
    input.service_info.metric[_] == "ResponseTime"
    some i, j
    contract.quality.services[i].ResponseTime.machinetypes[j] == "edge"
    input.metric.ResponseTime > contract.quality.services[i].ResponseTime.value
}
violation[[input.client_info, "Data quality violation"]]{
    input.service_info.metric[_] == "DataAccuracy"
    some i
    contract.quality.data[i].Accuracy.operators == "min"
    input.metric.DataAccuracy < contract.quality.data[i].Accuracy.value
}
```

QoA4ML specs and prototype: https://github.com/rdsea/QoA4ML
"QoA4ML - A Framework for Supporting Contracts in Machine Learning Services," ICWS *2021* doi: 10.1109/ICWS53863.2021.00066.

Example: monitoring utilities and observability service with QoA4ML

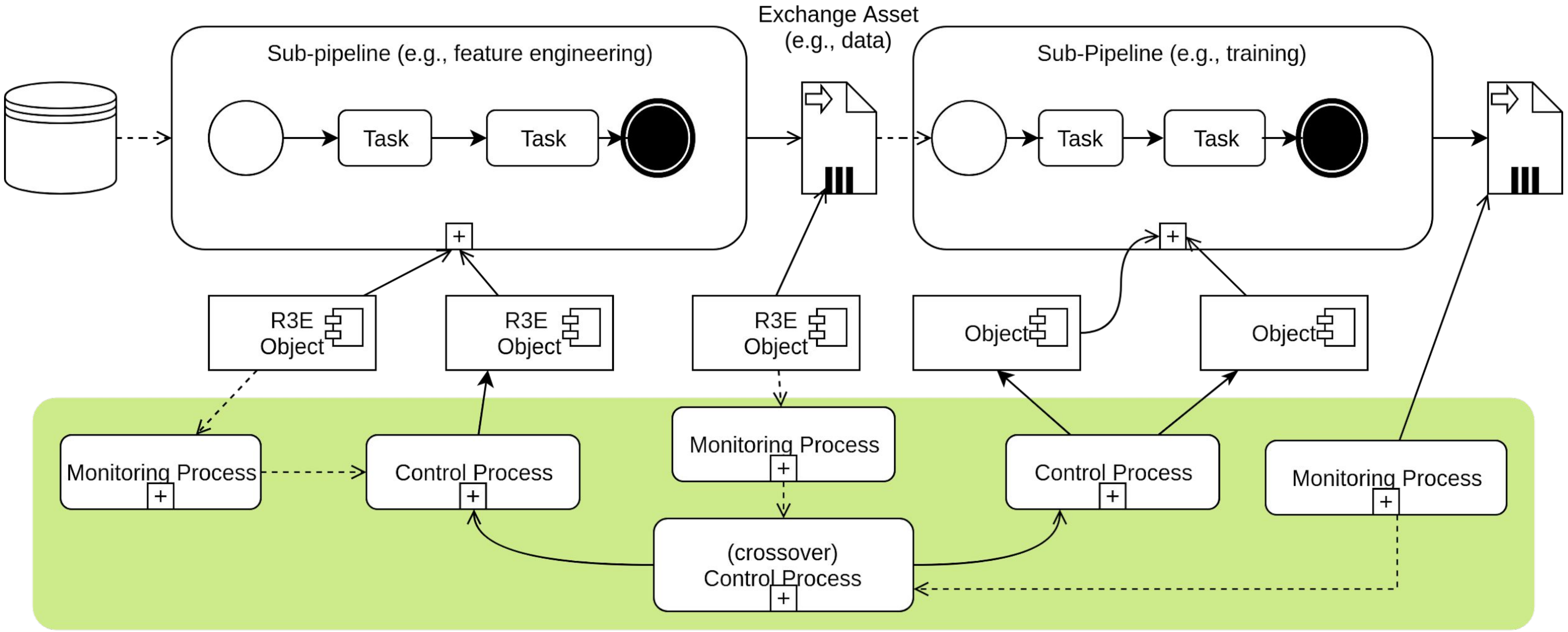
Monitor ML-specific attributes (and others)

Design for different engines to be used



Plugins, reuse well-known monitoring systems

Using control processes to ensure R3E constraints



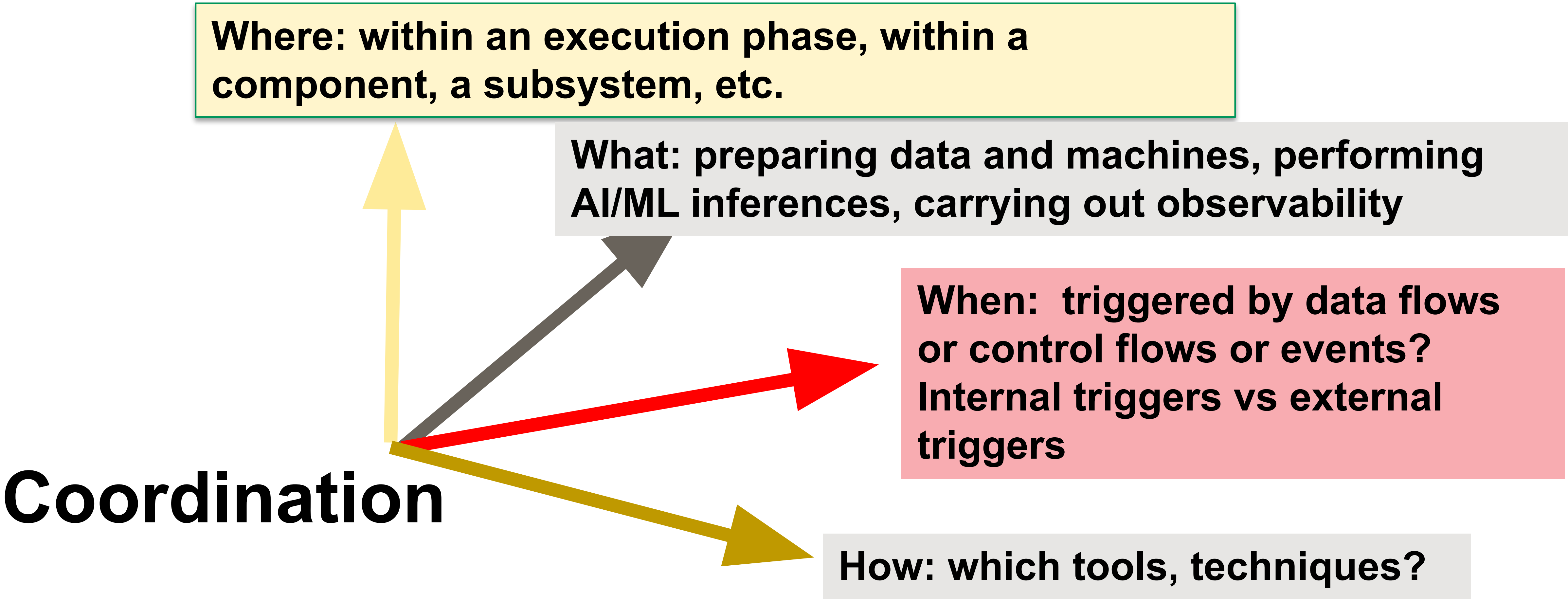
- R3EObject: encapsulate (sub)systems that can be controlled
- Control processes: can be complex with AI/ML-based algorithms

Coordination for R3E

Where do we need “coordination”

- Computing
  - scale and control data processing
  - dynamic serving: manage loads and services
- Time
  - between different sessions and overtaking
- Intelligence
  - switch and combine human+ai, elasticity of capabilities
- Coordination needs monitoring/observability
  - logging, tracing, monitoring of infrastructures, consumer requests and service/data tasks

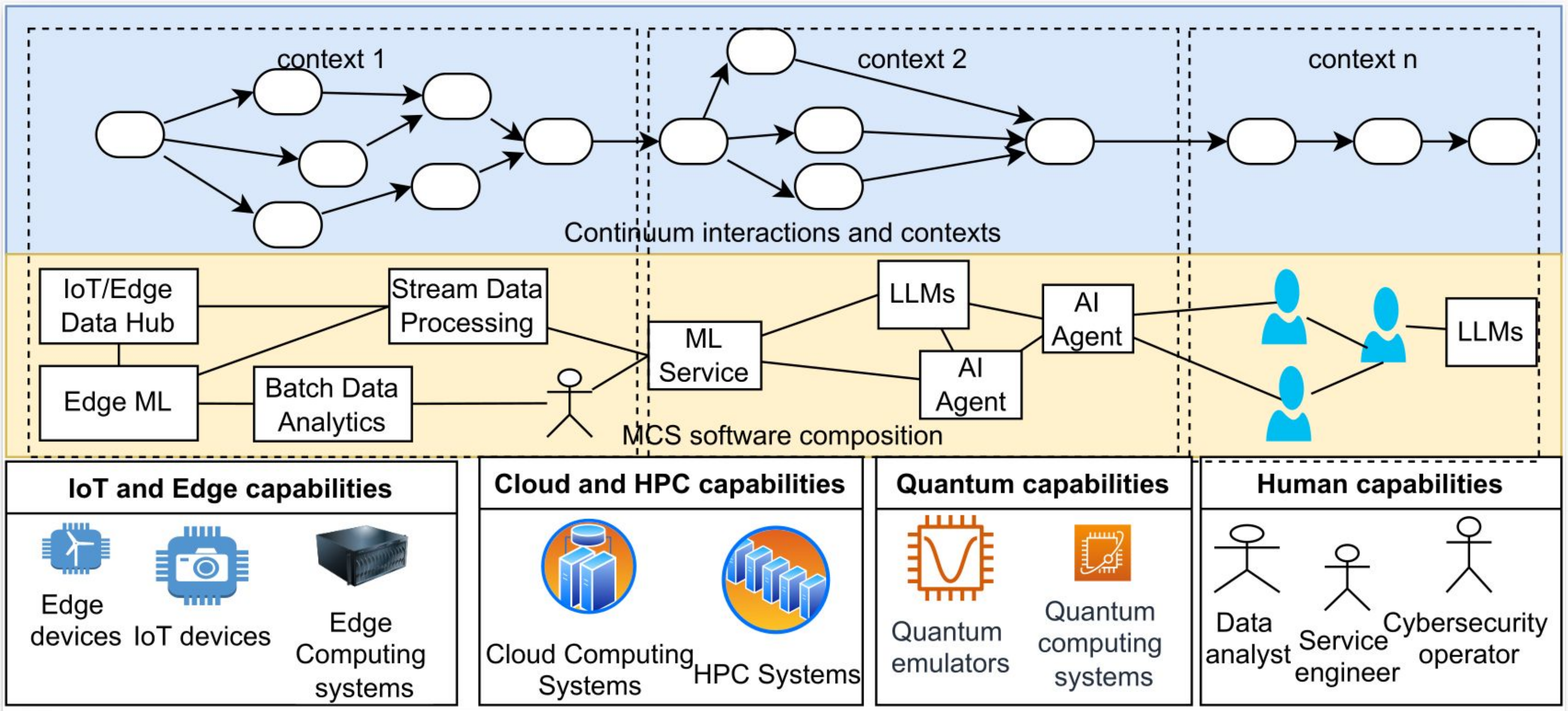
W3H: what, when, where and how for coordination



Different coordination architectures

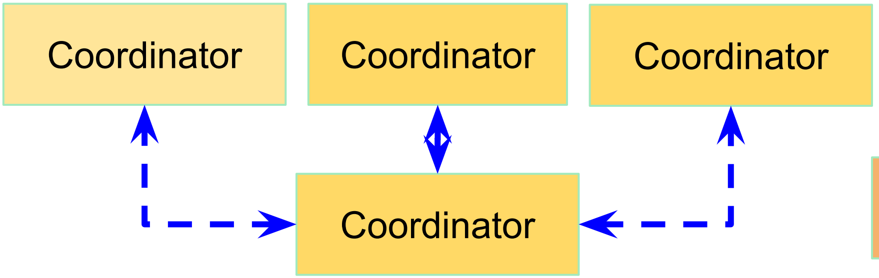
Capabilities in multi-continuum computing are from multiple (sub)systems/infrastructures

⇒ *system of system coordination*

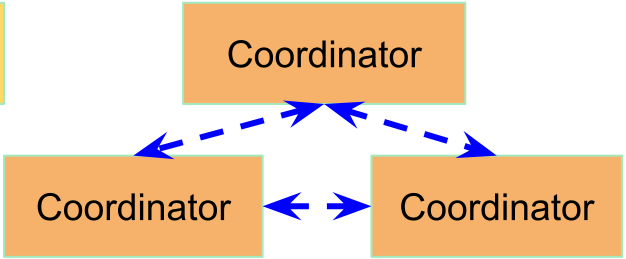


Possible designs

Single one-for-all coordination



Hierarchical coordination



Peer-to-peer coordination

Coordination styles

- Orchestration
    - task graphs and dependencies are based on control or data flows
    - dedicated orchestrator
        - tasks triggered based on completeness of other tasks or the availability of data
    - often implemented as workflows

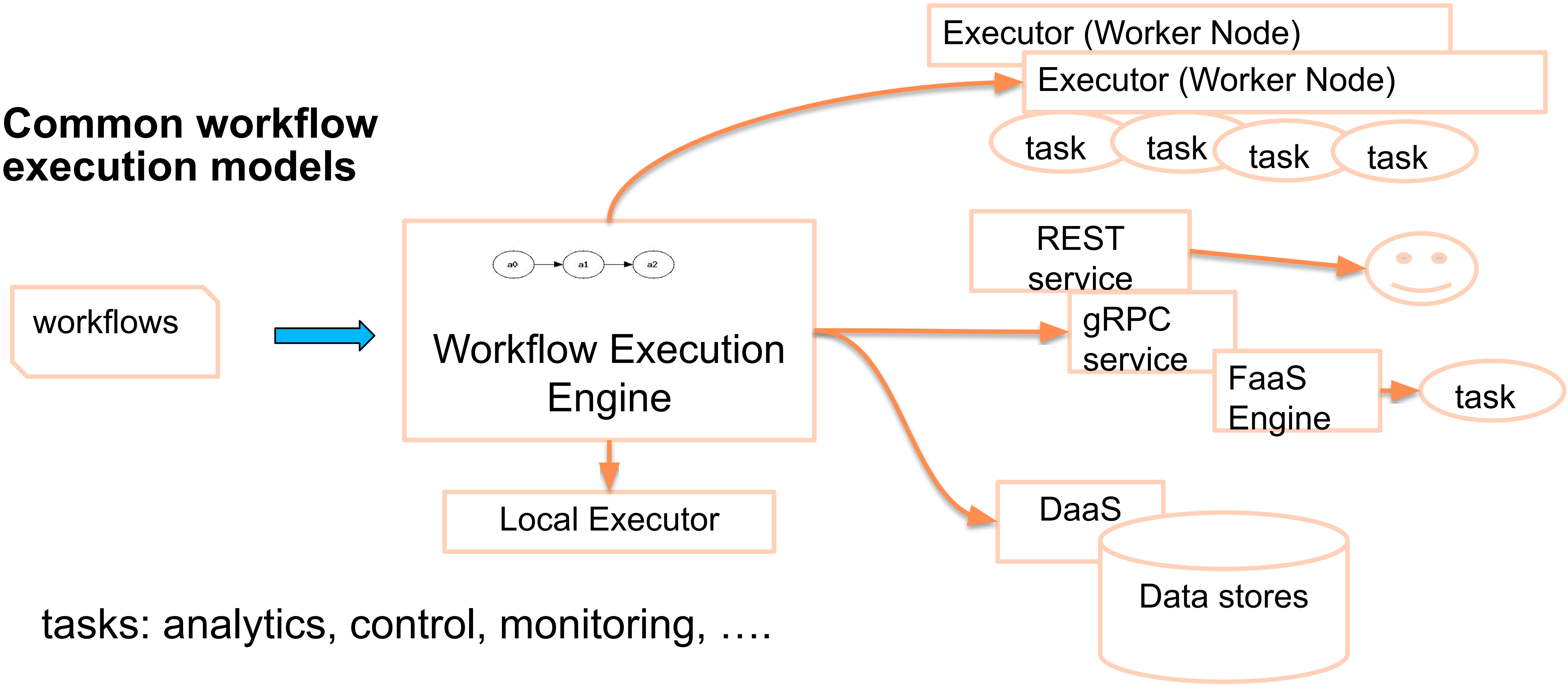
- Reactiveness/choreography
    - follow reactive model
        - tasks are reacted/triggered based on messages

Coordination using the orchestration architectural style (1)

- Using workflow
  - architectures are well-known, leverage many types of services and cloud technologies
- Required components
  - workflow/pipeline specifications/languages (also UI)
  - data and computing resource management, external services
  - orchestration engines (with different types of schedulers)
- Execution environments
  - cloud platforms (e.g., VMs, containers, Kubernetes)
  - heterogeneous computing resources (PC, servers, Beelink, etc.)

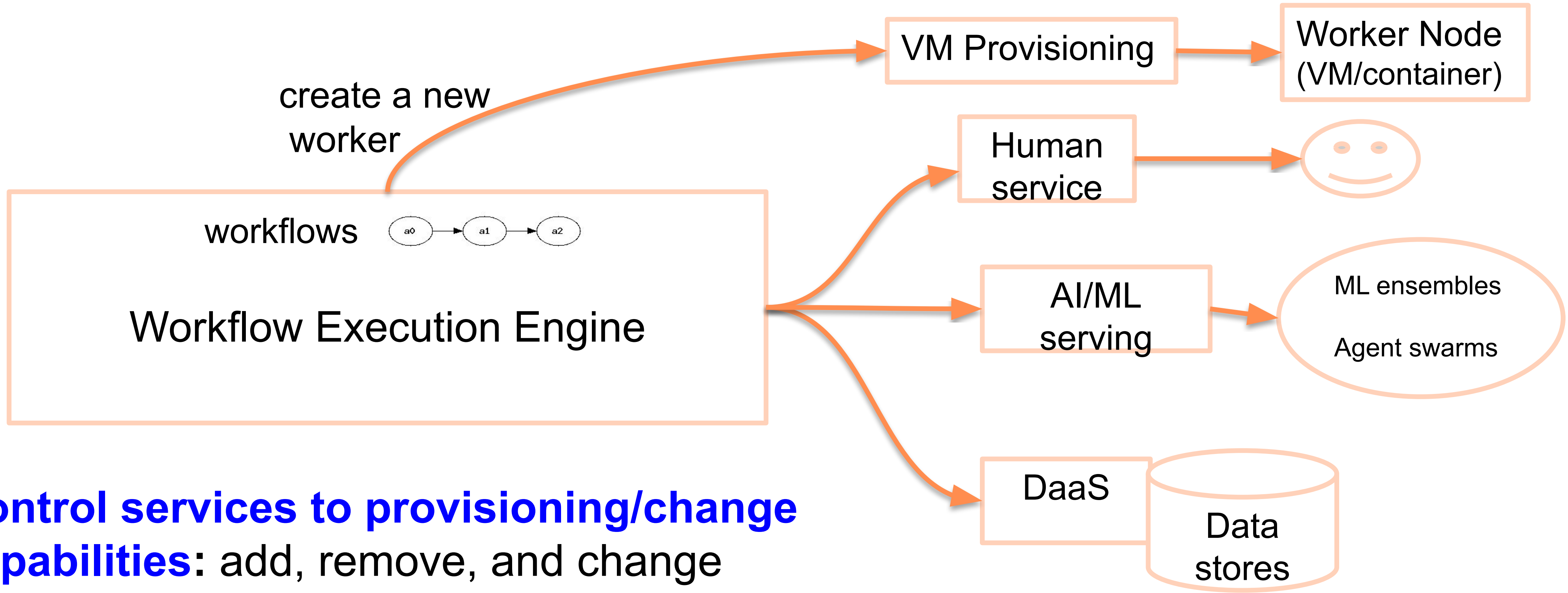
Coordination using the orchestration architectural style (2)

Common workflow execution models



tasks: analytics, control, monitoring, ….

Coordination using the orchestration architectural style (3)



Control services to provisioning/change capabilities: add, remove, and change capabilities

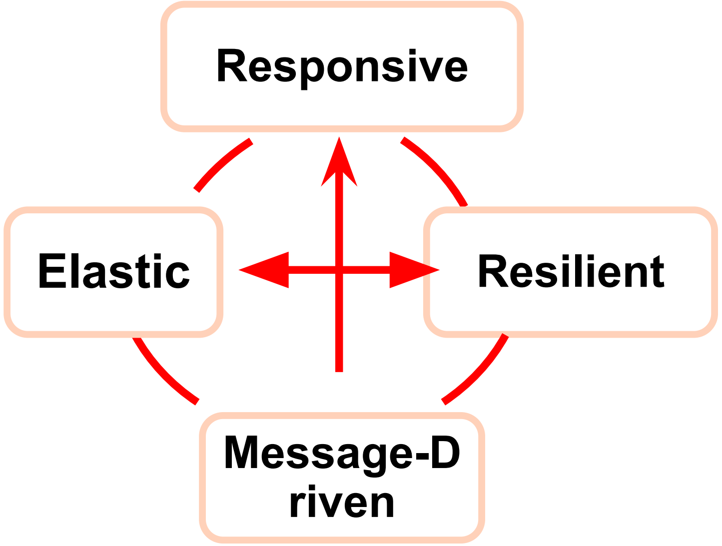
Coordination using the orchestration architectural style (4)

- **Common workflow systems**
  - use to implement different tasks, such as data processing, machine provisioning, service calls, data retrieval, human tasks
  - examples: Airflow (https://airflow.apache.org), Uber Cadence (https://github.com/uber/cadence), N8N (https://n8n.io/)

- **Serverless-based workflows implemented in different tools**
  - E.g., Amazon Step Function, Alibaba Cloud Serverless Workflow, CNCF Serverless Workflow

Coordination with the choreography/reactiveness style (1)

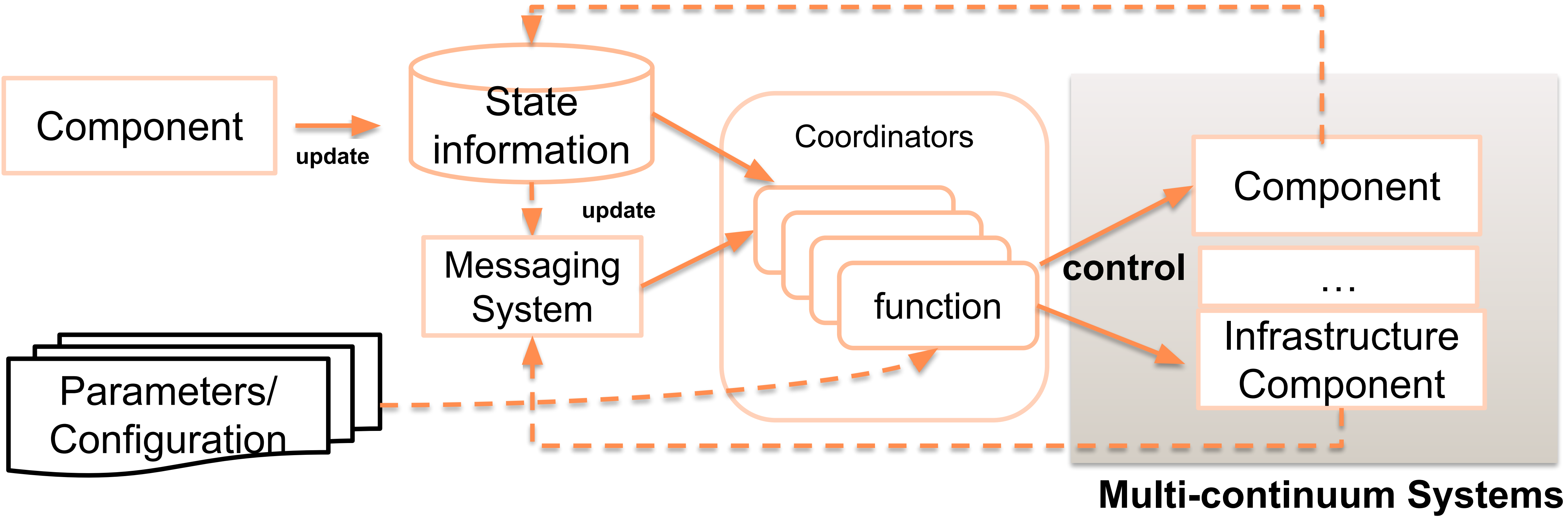
Reactive systems



Source: https://www.reactivemanifesto.org/

- Messaging systems
  - Kafka, RabbitMQ, ZeroMQ, Amazon SQS, …
  - types of messages and semantics must be defined clearly
- Triggers and controls
  - the serverless/function-as-a-service model: trigger a function/task based on a message
    - AWS Lambda, Google Cloud Function, Knative, OpenFaaS, Azure Functions
  - the worker model:
    - light weighted microservices and job workers listening messages to trigger (remote) functions/tasks

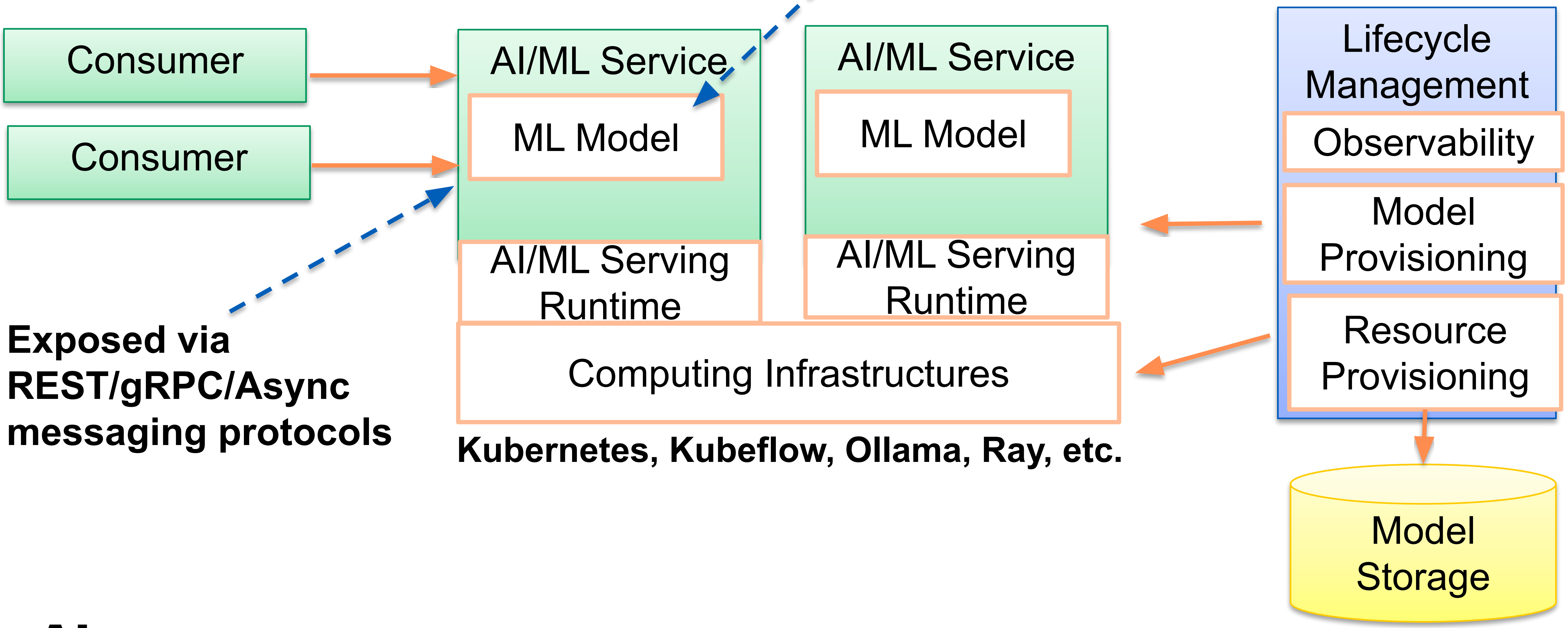
Coordination with the choreography/reactiveness style (2)



Some patterns for coordinating R3E in AI/ML services

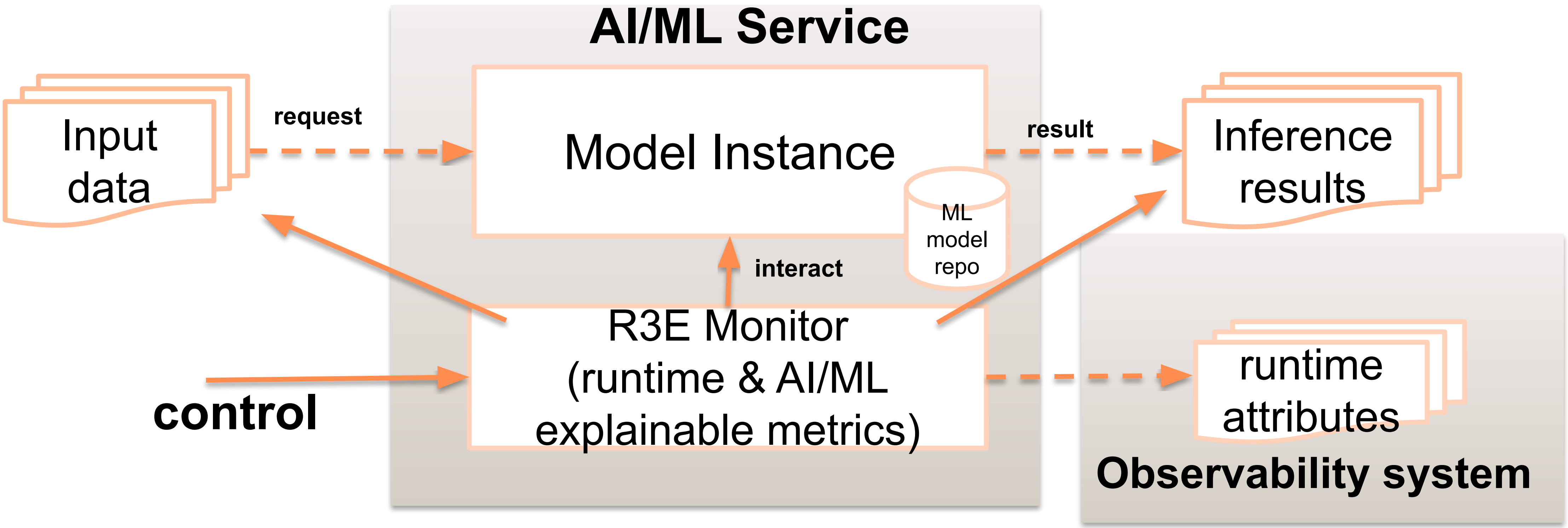
AI/ML Service Serving design

Different integration techniques (containerization, external process execution, in-process execution)



R3E runtime attributes

Capture important metrics



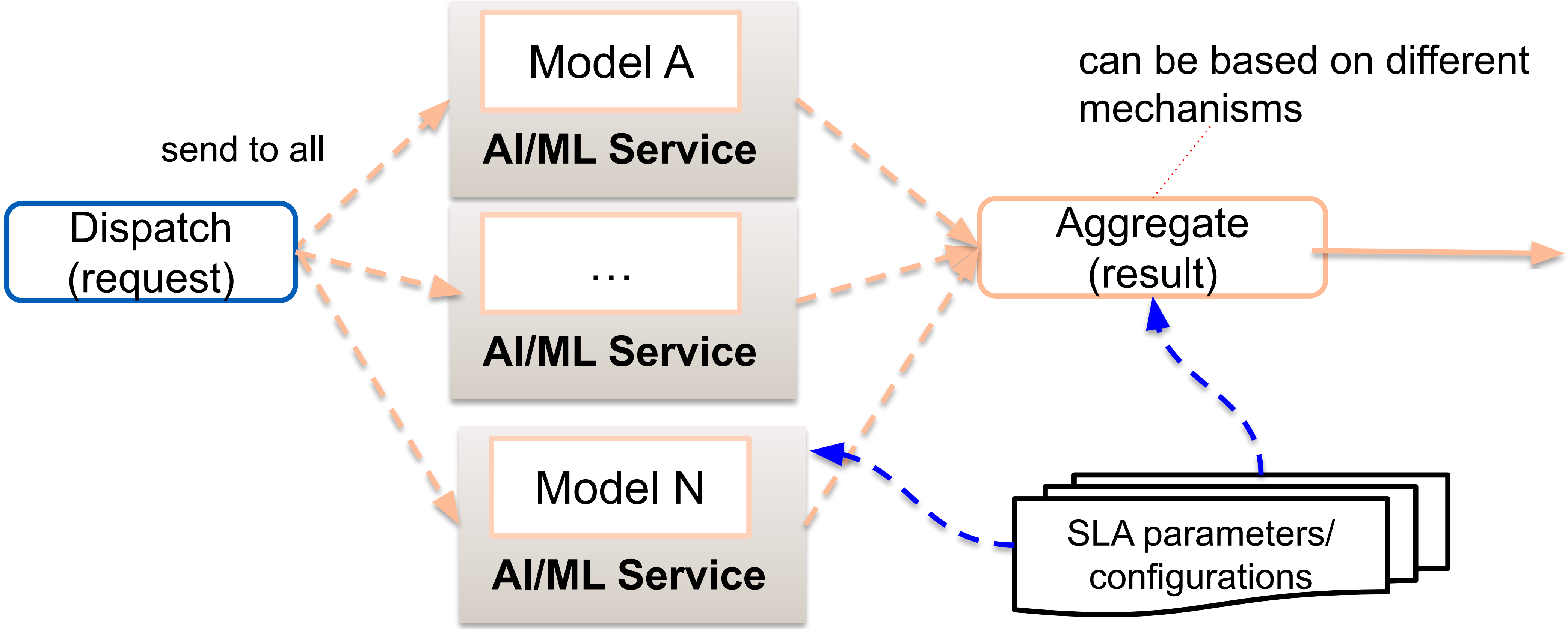
Optimizing R3E

- Serving platforms/toolkits:
    - Ray, BentoML, Seldon, KServe, Ollama, etc.
    - Also Nvidia Triton, AMD Inference, etc., serving runtime
- Modes, e.g.
    - batch serving, autoscaling, asynchronous serving
- Varying parameters, e.g,
    - batch serving (batch size, timeout, latency/response)
    - resources and autoscaling (replicas, CPU/GPU, memory)
    - queuing (concurrent requests)

⇒ many ways for optimizing R3E in serving!

Composition - Ensemble

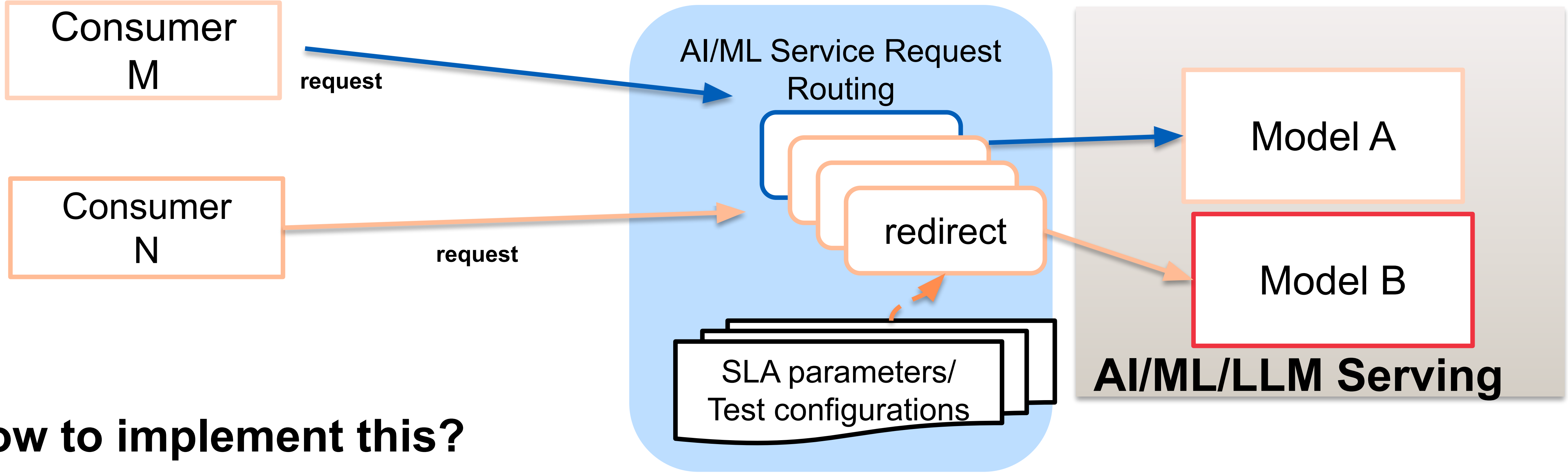
Ensembles of different models with different qualities/SLAs



Also for AI Agents/Human services

SLA-based serving

Different AI/ML models with different qualities/SLAs



How to implement this?

Amazon Sagemaker Example: https://docs.aws.amazon.com/sagemaker/latest/dg/model-ab-testing.html

Load balancing/scaling model serving

- Leveraging common load balancing techniques (queues, gateway, etc.)
- AI/ML inferencing capability by an AI/ML model is encapsulated into a (micro)service or a task
- As a service
  - with well-defined APIs (e.g., REST, gRPC), e.g., Dockerized service
  - using load balancing, gateway and orchestration techniques
- As a task
  - using workflow management techniques to trigger new tasks
  - support scheduling, failure management and performance optimization by leveraging batch processing techniques

Study log for this week

Think about

- What does it mean R3E for *YOUR services/systems in multi-continuum environment*?
- Read one of the papers in **the reading list** for today's lecture

Robustness, Reliability, Resilience and Elasticity

Then

- in your experience/work, which ones of R3E concern you most? Why? What would you do? What do you look for?
- ~1-2 page – submit it to the MyCourses for comments/feedback (keep it in your git)

Kiitos

aalto.fi