
This is an electronic reprint of the original article.
This reprint may differ from the original in pagination and typographic detail.

Truong, Linh

Coordination-aware assurance for end-to-end machine learning systems: the R3E approach

Published in:

AI Assurance: Towards Trustworthy, Explainable, Safe, and Ethical AI

DOI:

[10.1016/B978-0-32-391919-7.00024-X](https://doi.org/10.1016/B978-0-32-391919-7.00024-X)

Published: 01/01/2022

Document Version

Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

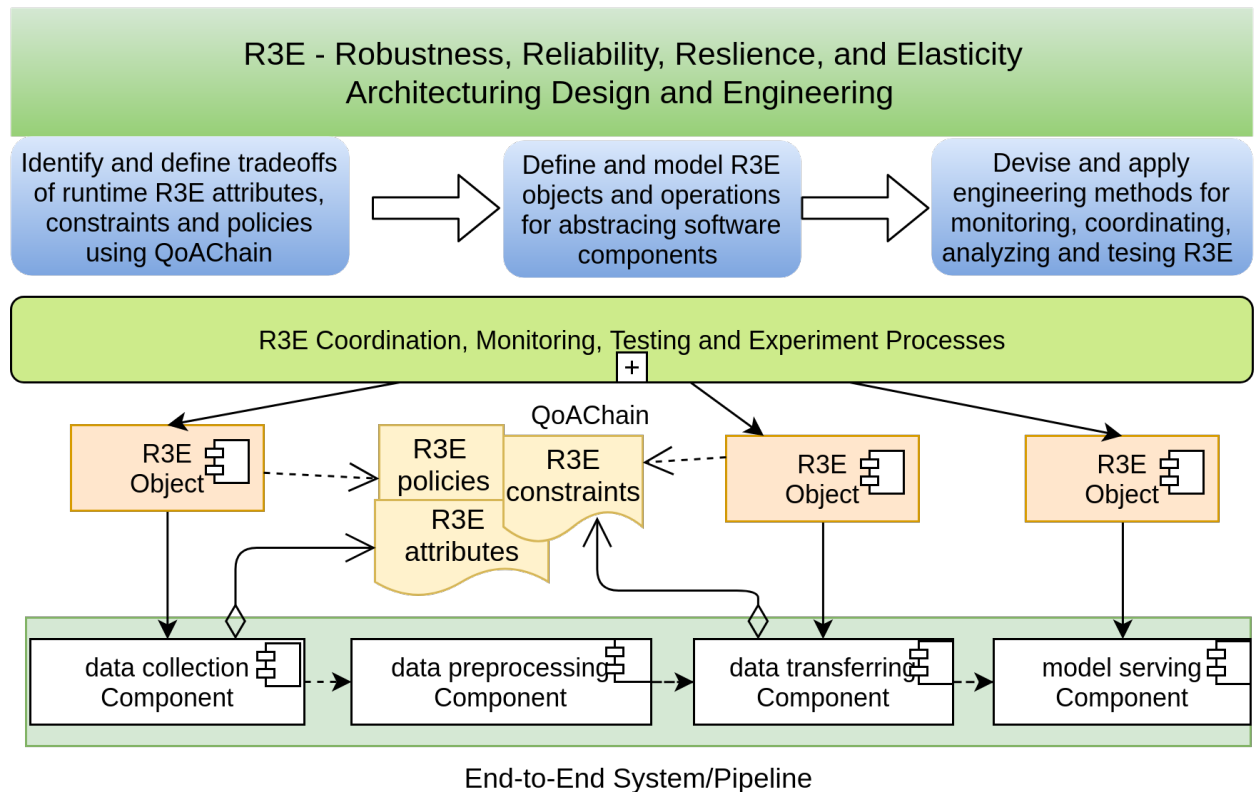
Please cite the original version:

Truong, L. (2022). Coordination-aware assurance for end-to-end machine learning systems: the R3E approach. In F. Batareseh, & L. Freeman (Eds.), *AI Assurance: Towards Trustworthy, Explainable, Safe, and Ethical AI* (pp. 339-367). Elsevier. <https://doi.org/10.1016/B978-0-32-391919-7.00024-X>

Graphical Abstract

Coordination-ware Assurance for End-to-End Machine Learning Systems: the R3E Approach

Hong-Linh Truong



Highlights

Coordination-ware Assurance for End-to-End Machine Learning Systems: the R3E Approach

Hong-Linh Truong

- This chapter characterizes robustness, reliability, resilience, and elasticity (R3E) in architectural designs for end-to-end big data machine learning systems.
- We provide a novel model of quality of analytics chain (QoAChain) to abstract and define constraints for assuring robustness, reliability, resilience, and elasticity of end-to-end machine learning.
- We present a concept of R3E objects and operations abstracting components in big data machine learning systems.
- We discuss engineering methods for coordinating, monitoring, analyzing, and testing R3E attributes.

ARTICLE INFO

Keywords:

software systems
machine learning
big data
software architecture
elasticity
cloud computing
engineering analytics


ABSTRACT

Concerns of robustness, reliability, resilience, and elasticity in Machine Learning (ML) systems are important and they must be considered in trade-off with efficiency factors. However, they need to be supported and optimized in an end-to-end manner, not just for ML models. In this chapter we present a conceptual approach to architectural design and engineering of the robustness, reliability, resilience, and elasticity (R3E) for end-to-end big data ML systems at runtime. We propose quality of analytics as a contractual means for optimizing end-to-end big data machine learning (*BDML*) systems. Based on that, we propose to define and abstract diverse types of components under R3E objects and devise operations and metrics for managing R3E attributes. Through a set of proposed coordination, monitoring, analytics and testing methods, we identify essential tasks for tackling R3E concerns when developing *BDML* systems. Finally, we illustrate our approach with an example of an end-to-end *BDML* system for building objects classifications.

1. Introduction

Big Data Machine Learning (*BDML*) systems enable different types of ML-based pipelines, which deal with big data in motion or at rest. End-to-end *BDML* systems support tasks from processing raw data to producing inference results. Thus, *BDML* systems involve several different software components, including data sources collectors/connectors, message brokers, edge data preprocessing and aggregators, cloud data stores, ML serving platforms, and ML services. These components are cross-layered and cross-infrastructural, due to the nature of diverse ML pipelines and data to be supported by such systems. Thus, components of an end-to-end *BDML* system are potentially deployed and offered in multiple edge and cloud infrastructures. Typically, the data to be inferenced and the application using the ML model-as-a-service are from the consumer, whereas the ML model-as-a-service can be run in the edge or cloud by the ML service provider, which offers the service to many consumers. Furthermore, computing, storage and communication services might be offered by other providers. Such systems for real-world ML must be designed with robustness, reliability, resilience and elasticity (R3E) concerns from a multi-party perspective. Although individual components may be designed and tested with certain degrees of R3E, the challenging question for the development of end-to-end *BDML* systems is to guarantee expected runtime R3E attributes across layers and infrastructures. Therefore, recently, the role of software systems and underlying distributed computing platforms and their intersections with ML have been discussed intensively. Since *BDML* systems are complex and typically used for critical businesses, the R3E attributes play a key role in *BDML* software architectures and implementations. Ensuring R3E is challenging for complex software systems because R3E attributes are highly interdependent and multi-dimensional. Especially, R3E attributes in *BDML* systems are related to three aspects: *services*, *data*, and *ML models*. The key research question in our work is how to build and optimize the R3E attributes for *BDML* systems in an end-to-end manner; and the “end-to-end” aspect forces us to examine various components of *BDML* systems together, following their dependencies, interactions and functions.

This chapter presents a novel conceptual approach to R3E engineering for *BDML* systems, in which we will focus on software architecture and design aspects. We develop abstractions and methods for architectural designs, runtime optimization, and engineering analytics in *BDML*. Our approach considers different levels of abstractions of *BDML*, from data collection to training to model serving to determine key constraints, engineering steps, monitoring, and management processes for making *BDML* robust, reliable, resilient, and elastic. Our conceptual approach makes the following contributions:

 linh.truong@aalto.fi (H. Truong)
ORCID(s): 0000-0003-1465-9722 (H. Truong)

- QoAChain as a means to combine quality of analytics constraints, services contracts, and data contracts for specifying runtime R3E attributes and constraints.
- abstractions and models for R3E objects and operations in *BDML*.
- engineering methods for coordinating, monitoring, analyzing and testing R3E attributes.

Our approach covers key aspects of R3E engineering and layouts the foundational work for the development of specific techniques and tools to support R3E in *BDML* systems. To illustrate our approach, we will use a realistic example of end-to-end *BDML* for building objects classifications.

The rest of this chapter is organized as follows: Section 2 characterizes R3E in *BDML* and presents our motivating examples and research questions. Section 3 presents our R3E approach. We present a concrete example of R3E aspects and identified requirements in Section 4. Further related work will be discussed in Section 5. We conclude the chapter and outline the future work in Section 6.

2. Background and Motivation

2.1. Background – Characterizing *BDML*

A *BDML* system can be characterized as follows:

- *system structures and functions*: a *BDML* system includes various components implementing different functions. Examples of components are a data storage service and an ML serving platform, whose functions are storing data and serving ML models, respectively. Components have different relationships and possible inputs and outputs. R3E attributes can be associated with individual components, a set of components, and the system as a whole.
- *supporting computing, data and communication infrastructures*: computing infrastructures provide different types of computing resources for different tasks, notably data preprocessing, training, and serving. Typically, such infrastructures include advanced computing systems like CPU/GPU resources, containers and Kubernetes, message brokers, and edge systems. The data infrastructures provide data for training and data being inferenced as well as other types of data related to ML, such as ML model experiments and performance of ML services. Our focus is on big data infrastructures.
- *runtime quality/capabilities*: they include multiple attributes, for example, regarding to fault-tolerance, high performance, high availability, and security of software services. From the data view, a *BDML* system has to deal with big data characteristics, such as volume, variety, velocity, and veracity, from the data source to the end of the ML pipelines. Furthermore, ML models have different quality attributes, depending on domain requirements and business contexts.

For example, Figure 1 shows a view from common tasks in end-to-end *BDML* pipelines. From the ML pipelines perspective, the system-as-a-whole can be seen as a meta pipeline orchestrating different sub systems, where each subsystem can be implemented differently, such as with Airflow, Lambda, TensorFlow, and other supporting services. Each of them requires a variety of components for data, software services, ML algorithms, and pipeline orchestration.

In this chapter, we consider well-studied R3E attributes in the state-of-the-art literature:

- robustness attribute (Gribble, 2001; Laranjeiro et al., 2021) is about the ability to cope with errors, such as with the error of the data (Sehwag et al., 2019).
- reliability attribute (Littlewood and Strigini, 2000; Saria and Subbaswamy, 2019; Elsayed, 2012) is about the ability to properly function/operate according to the service specification, e.g., the availability of a service must be 99%.
- resilience attribute (Trivedi et al., 2009; Brtis et al., 2021) is about the ability to hold out required capabilities under adversity, e.g., due to system failures or security attacks.
- elasticity attribute (Dustdar et al., 2011) is about the ability to stretch and return to normal service capabilities, e.g., under external forces of usage demands.

We will rely on common definitions and usages of these attributes from the big data and ML perspectives. Table 1 gives examples about key R3E concerns and factors from big data and ML views of *BDML* systems. Our approach will support R3E attributes in such common senses.

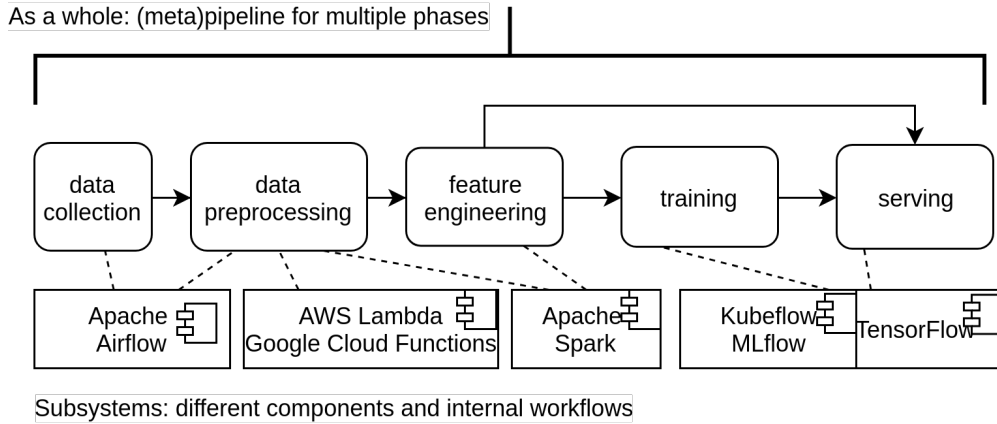


Figure 1: Example of ML pipelines and components in *BDML* systems

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

Table 1

Common R3E with big data and ML concerns

2.2. Motivating example – machine learning for classifying building elements

We consider a prototype for an end-to-end *BDML* system for classification of Building Information Modeling (BIM) elements in the architect, engineering and construction domain. ML-based BIM classification allows speedup the design and check conformity of building models. In our collaboration, an initial end-to-end BIM *BDML* system has been developed using various AWS services for moving data and ML capabilities are built with TensorFlow and Keras (Ryu et al., 2021). Figure 2 shows a simplified view of a new architectural design for the discussion of the role of R3E in this chapter, where we are leveraging serverless platforms to better manage and optimize the complex relationships between various components. In the new design, data exported from user tools will be moved to *Data Service*. New data will be detected, and *preprocessing* and *feature engineering* will be triggered by serverless platforms, before *ML Service* serves requests of classifications. Atop the *ML Serving Platform* we have *ML Service* with different *ML Models* for BIM classification.

Figure 2 not only shows an end-to-end system with various components but also creates clear interfaces between different sub-pipelines, like *preprocessing*, *feature engineering* and *serving*, enabling us to carry out different

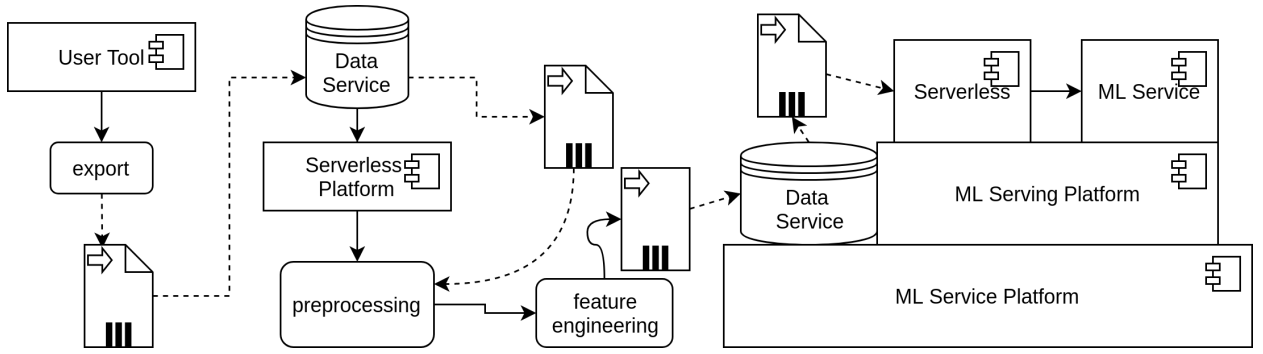


Figure 2: Overview of an end-to-end big data machine learning system for BIM

performance/cost optimizations for different pipelines and their underlying components. It also allows us to deal with R3E attributes more flexible for subpipelines and underlying components. However, it creates various R3E concerns that need to be addressed together. For example, *ML Service* has to be elastic to support different requirements with respect to accuracy, cost, and performance. This is dependent on the elasticity of the underlying *ML Service Platform*, which is strongly linked to computing resources, and on the output of *feature engineering*, which in turn is strongly dependent on the exported data sent to *Data Service* and *preprocessing* robustness and reliability.

2.3. Research questions

Key issues of end-to-end *BDML* are not just about efficiency, such as highly responsive in serving the classification with a minimum cost, but also in trade-offs with robustness, reliability, resilience, and elasticity. For example, in the BIM scenario (Section 2.2), the accuracy of inference results from *ML service* and the resilience of *ML service* are more important than the response time due to the business nature of the domain. As recognized in (Ackley, 2013), performance must also be aligned with robustness and resilience. The robustness of the ML model depends on the data input. From the computation and network, reliability concerns for edge-cloud have many issues (Suryavansh et al., 2019; Nguyen et al., 2019). The reliability concern from the data aspect is that the data source must provide “reliable data”, interpreted as the data quality and quantity satisfied the required conditions. On the other hand, from the service viewpoint, the ML model serving will be considered as a reliable service when it can return the results in specified time. This turns out to be dependent on multiple factors, such as, the reliability of the underlying computing resources (e.g., no failure) and the elasticity of the resources (e.g., in order to assure response times in the expected range).

We see that R3E concerns exist in different parts of a *BDML* system. However, currently, there is no systematically way to capture, represent, monitor, and optimize such R3E attributes from the design and architecture viewpoint. Our vision in this chapter is:

R3E attributes can be systematically modeled, programmed, and captured at different levels of abstractions in *BDML* systems, enabling the coordinated optimization of these attributes in an end-to-end view, based on specific contexts of the intended end-to-end ML pipelines executed in *BDML* systems.

Consequently, we have the following important research questions (RQs):

- RQ1: *what would be the model for abstracting R3E constraints?* With diverse R3E concerns, we need to capture key R3E attributes and describe them into appropriate constraints.
- RQ2: *how can we abstract complex components in the R3E view and define suitable operations for managing R3E?* Components in *BDML* systems need to be managed through the R3E view, which should capture attributes and essential operations.
- RQ3: *which are the key engineering methods for achieving R3E?* Engineering methods for monitoring and managing R3E attributes across components of *BDML* systems must be laid out, paving the way to develop suitable tools and frameworks.

The R3E approach will provide key conceptual steps and components to address the above-mentioned questions.

3. Key elements of R3E Approach

3.1. QoAChain – Chaining diverse types of quality constraints as a contract for optimizing end-to-end *BDML*

RQ1 requires us to determine how to abstract R3E attributes and to specify R3E concerns for optimizing *BDML* systems. Figure 3 gives a high-level view of the complex relationships among various concerns of different attributes when optimizing *BDML*. Given an application, big data and ML pipelines are combined and executed to analyze input data (data in) and produce results. Such executions are carried out with edge-cloud resources as services. There are many questions with respect to attributes associated with used models, input data, results, and execution environments of computing, data and communication services, as exemplified in Figure 3. These concerns are from different involved stakeholders, such as the application users, the *BDML* system provider, the developer and scientist of ML pipelines, and the resources provider. Overall, they reflect the concerns of dealing with trade-offs between R3E and efficiency.

Consider the complex relationships among various components and stakeholders in *BDML*, we choose to combine the concept of quality of analytics (QoA) (Truong et al., 2018), machine learning service contracts (Truong and Nguyen, 2021), and data contracts (Balint and Truong, 2017; Truong et al., 2012) for end-to-end *BDML*. We summarize these works in the following:

- Quality of Analytics (QoA) (Truong et al., 2018) emphasizes the need to optimize data analytics based on specific contexts that is elastic. It characterizes complex relationships between quality of results, performance, and cost that are not fixed, but changing according to requirements, even for the same system:
 - Quality of results, outputted from data analysis tasks including ML ones, are characterized by the user/domain expert, e.g., quality of data of the output and the accuracy of predictions.
 - Input data has complex characteristics with respect to, for example, quality of data and data volume and velocity, that strongly influence infrastructural resources as services, such as task execution, computing machines, and storage.
 - Complex types of cost (money) and performance are based on business purposes, contextually expected and changed by involved stakeholders.
- The recent work on machine learning contracts (Truong and Nguyen, 2021) defines contractual terms between ML service providers and ML customers. ML contracts focus on ML-specific attributes, such inference accuracy, at runtime that are agreed between the customers and the services.
- Existing data contracts (Truong et al., 2012; Balint and Truong, 2017) focus on constraints on data to be delivered from data sources (providers) to consumers. They focus very much on quality of data attributes.

Clearly the above-mentioned concepts aim at guaranteeing important constraints seen in *BDML* systems. ML-specific attributes, data quality attributes, and common service attributes can be associated with various parts of a *BDML* system. The associations can be for individual components or a whole pipeline, and can indicate different expectations in the *BDML* system. Due to the diversity of component types, inputs and outputs, it is difficult to have a single way to specify such constraints for *BDML* systems.

A “reliable *BDML* system” should guarantee the specified runtime quality attributes built from the work on QoA, ML contracts and data contracts, while maintaining designed R3E attributes. Due to the nature of ML systems, we can use these concepts to specify constraints for different parts of a *BDML* system for different purposes, such as:

- data contract: a constraint on data completeness for input IoT data.
- ML contract: a constraint on inference accuracy for inference results.
- common service contract: a constraint on the response time for the end-to-end processing.

These examples show that runtime constraints can be defined for different components for different attributes and these constraints might be specified by different models. An *BDML* system is designed and optimized for different ML pipelines, which serve different business purposes, depending on the usage of the resulting outcome of the pipelines

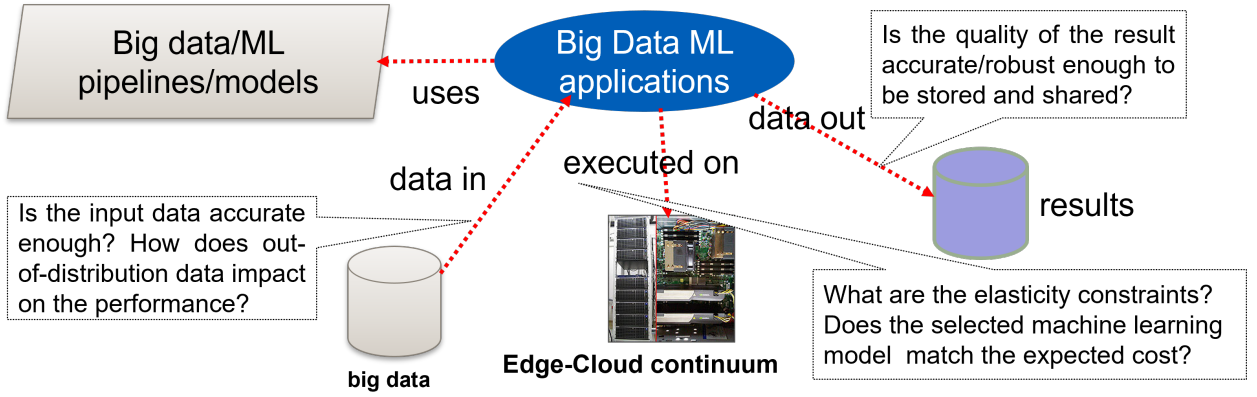


Figure 3: Concerns among components and stakeholders in optimizing ML

and the business goal of the provider of the pipelines supported by the *BDML* system. Therefore, a QoA-based approach can help to deal with the diversity of what, when, where, and how runtime attributes related to R3E can be supported. The QoA-based approach should include metrics for services, data and ML models to reflect the end-to-end view. It should support human-in-the-loop and domain expert integration when defining QoA, due to the domain aspect of end-to-end *BDML*. To this end, we define “Chaining QoA for *BDML*” (QoAChain) as a contractual means for optimizing end-to-end *BDML* systems. QoAChain constraints described in a “contract” for optimizing R3E attributes (i) implement service contract and data contract models, (ii) enable monitoring and optimization techniques centered around contracts, and (iii) allow runtime changes and updates according ML-specific contexts by people or intelligent software. QoAChain constraints are based on various metrics inherent in *BDML*.

Figure 4 shows key sub-elements of a proposed QoAChain and its relation to a *BDML* system. First, in our view, a *BDML* System consists of many Components; each Component may have sub components. A Component will utilize some resources (in order to implement required functions) and/or will deliver resources (e.g., featuring data/resulting prediction). Resources in our view can be simple or complex, and they are not just infrastructural resources. There main categories of resources to be utilized or delivered are Services and Data. Services can be further divided into different types, such as for data processing, computing, storage and inferencing. Data can be used to represent input data as well as output data (e.g., inference result in the case of ML service). In terms of quality, represented by Quality, there are many attributes known in big data and ML, such as ResponseTime, Data Quality, and Inference Quality (see also Section 2.1) that we just illustrate some of them in Figure 4. QoAChain for a *BDML* System consists of different QoAConstraints, which are associated with Components or the *BDML* System as a whole. QoAConstraints are used to specify constraints on attributes that should be monitored and optimized for R3E. They include tradeoffs among Resources, Quality, and Costs. QoAConstraints can be implemented by using existing, specific contract specifications. Examples of constraints in a chain are:

- a constraint on data completeness for IoT input data sent to a message broker, which passes the data to an ML service for dynamic inference of the IoT data.
- a constraint on inference accuracy for an ML service, given a constraint on data completeness and data volume that the ML service handles in a window of time.
- a constraint on the response time between from the time a component sends a batch of data to a message broker to an ML service until the time the component receives the inference result.

Based on QoAChain, the next question is how to manage R3E attributes across multiple contexts in end-to-end *BDML* systems, such as to which components we should associate QoAChain and how to manage them.

3.2. R3E Objects and Operations

For **RQ2**, we will address fundamental abstractions for objects and operations for R3E. Consider the internal structure of a *BDML*: $BDML = \{c_1, c_2, \dots, c_n\}$ whereas c_i is a component, which is a part of *BDML*. A component

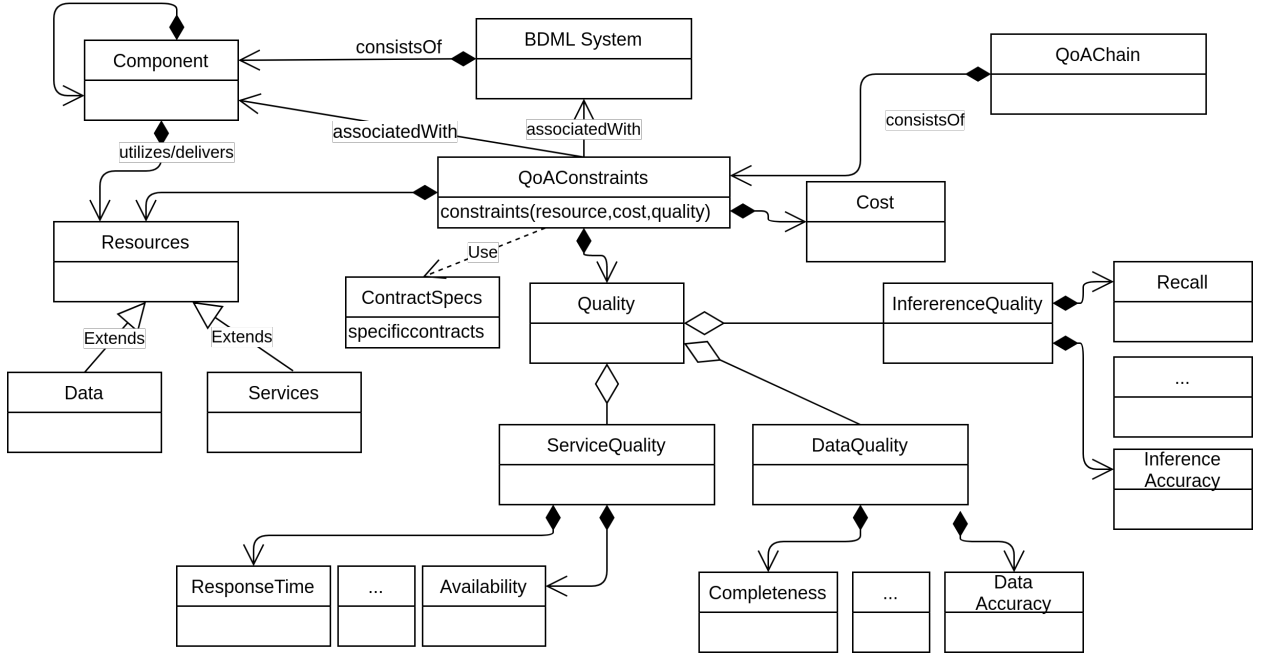


Figure 4: A simplified view of QoAChain and its relations to *BDML* systems and existing contracts

can be a software service, a container instance, a virtual machine (VM), or a middleware; a component can be instantiated as a resource-as-a-service. A component can be composed from a set of components, creating a complex component as a subsystem of a *BDML* system. For example, a subsystem for data preprocessing in a *BDML* can include containers and workflow orchestration components. Given the structure of *BDML* explained in Section 2.1, a component of *BDML* can be described using a set of objects; an object can represent a very complex component, such as an extract-transform-load process that filters data suitable for feature engineering, or represent a simple task, e.g., a data validation task.

3.2.1. Conceptualize R3E objects

In terms of management, we view components, pipelines, tasks as well as their input/output as programmable *objects*. We define an object as an *R3E object* if we can associate R3E policies and attributes with the object, meaning that we can examine R3E capabilities for the object and control these attributes. Given a *BDML*, not all the objects can be an R3E object. Furthermore, in the view of the developer, they might not see an object as an R3E object if they cannot apply R3E techniques. However, the operator of *BDML* might see that object as an R3E one. For example, consider an ML model which has no elastic parameters to influence robustness. The developer might not focus on the ML model as an R3E object. The operator sees that the underlying computing resources can be changed for the ML model, thus it can be an R3E object.

We propose to conceptualize R3E objects, shown in Figure 5. An *R3EObject* represents a *BDML Component*. *BDML Component* can be classified according to their functionality and layers, such as infrastructural objects, ML algorithm objects, and data objects. Furthermore, a *BDML Component* can be composed from other *BDML Components*. Therefore, we have a similar classification of *R3EObject*. An *R3EObject* will have to implement a set of operations/APIs for controlling and monitoring and will be associated with a set of attributes – *R3EAttributes*, each attribute is represented as a metric name and value.

When applying the R3E approach, R3E objects can be identified and built from two perspectives: existing knowledge about contemporary objects that we use, such as containers, VMs, and middleware, which already have built-in features for controlling certain aspects of R3E. Second, the newly objects to be developed for *BDML* must implement such features. Given an *BDML* system, in our approach, we do not need to represent *R3EObject* for all possible *BDML Components*. However, using a composition model, we can also build an *R3EObject* for the entire

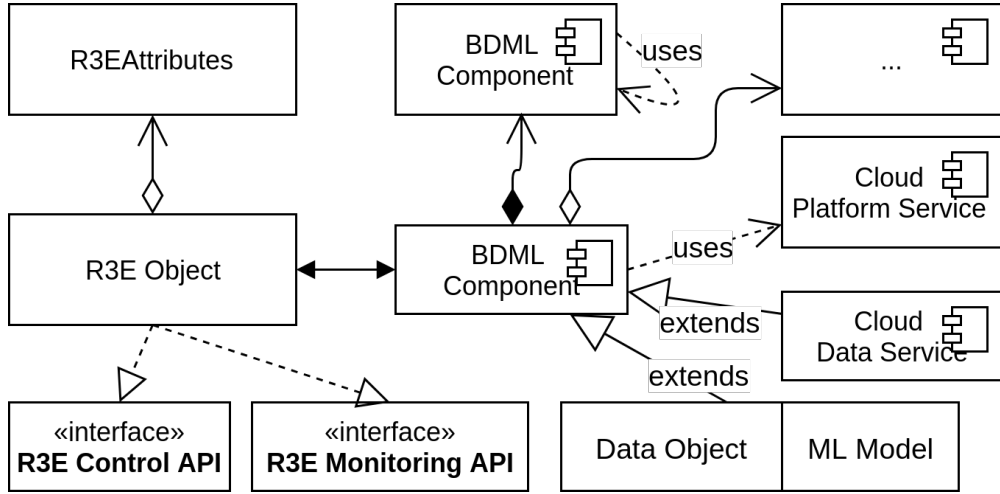


Figure 5: A simplified conceptual model of R3E objects

BDML system that links to other *R3EObject*s representing other components. Via the dependency of *R3EObject*s, we can capture the whole picture of the system to be optimized. For the implementation, we are investigating two models: *R3EObject* as implemented as a resource of a microservice (an adaptor model) and as an interface implemented within components themselves.

3.2.2. R3E attributes associated with R3E objects

We classify attributes into different sub categories, associated with services, data, and ML models, shown in Figure 4. Each attribute is represented as a metric under a tuple (*name*, *value*).

- Services quality: covers different types of attributes for a variety of services, including infrastructural computing services, data storage, communication services, and platform services. Common quality attributes are well-known in literature, such as response time, availability, and MTBF.
- Data quality: covers data quality metrics, such as completeness, timeliness, currency, validity, format, accuracy, and data drift.
- ML models quality: includes known quality in ML models, such as accuracy, F1 Score, and MSE.

These metrics are captured for individual components and composite components as well as tasks of ML pipelines carried out atop such components.

3.2.3. R3E operations and APIs

Given an *R3E* object, we must be able to control it to meet *R3E* constraints, which are pre-defined or changed during runtime. For example, if parameters of an ML model as a *R3E* object can be controlled to affect the ML model, we can then optimize the ML model for different degrees of robustness, reliability, resilience, and elasticity. Similarly, if an object performing feature engineering can be tuned with different granularity of feature extraction and selection, then we can control the object to have different data quality values. Furthermore, to allow for controlling, we must be able to monitor and query states of *R3E* objects at runtime. This can be done directly through querying the object or indirectly through the monitoring systems. Shown in Figure 5, two types of key operations are for *R3E* controlling and monitoring. An *R3E* operation associated with *R3E* objects will be implement as an API. Inputs and outputs of the API are centered around metrics and constraints specified in *QoAChain*.

3.3. Engineering Methods

For **RQ3**, we propose a set of engineering methods for *R3E* coordination, monitoring and analytics, and testing, benchmarking and experiments. We will describe engineering methods but leave the implementation of tools and frameworks for such methods out of the scope of this chapter.

3.3.1. Coordination for R3E

Having R3E objects enables us to optimize the *BDML* system through control and reconfiguration of R3E attributes, thus leading to changes in components of the *BDML* system. Due to the complexity and structure of the *BDML* system, coordination of such controls and configurations is challenging.

Architectural styles for R3E coordination: To perform the coordination of controls and reconfigurations of various R3E objects, we must consider suitable architectural styles coupled with *BDML* systems. Most architectures for end-to-end *BDML* systems follow either the reactive style or the workflow style as the basic architectural style. Furthermore, due to the complexity of individual components, each component might also follow the workflow or reactive style. Basically,

- reactive style: the data/event from one task/component triggers the next action in the pipeline/system (Smith, 2018). This model usually fits very well with large-scale *BDML* systems
- workflow style: a workflow is used to control tasks/components in ML pipelines/systems. However, most systems focus on leveraging workflows in the training or inferencing.

We design our approach to work with the reactive system style. This will be aligned with *BDML* pipelines consisting of components across different layers and different infrastructures (e.g., edge and cloud) and different providers. Furthermore, using reactive models, we can intercept *BDML* systems at different places to create optimization and feedback channels to support R3E. Figure 6 present the high-level components view. A *BDML* system consists of different subsystems, such as *Computing and Data Platform*, *Data Processing Platform*, and *Serving Platform*. Each subsystem is complex and can be implemented with different technologies. ML tasks in ML pipelines are spread in these subsystems and they are coupled through reactive principles by using messages. Therefore, we do not need a global workflow system to orchestrate them but we can use a set of R3E Managements. Each R3E Management will interact with a subsystem using three interfaces: R3EPolicies are used to control subsystems, R3EAttributes are used to capture states and R3EConstraints specifies QoAChain. Among R3EManagements the reactive principle is also used to provide an end-to-end view of the whole system.

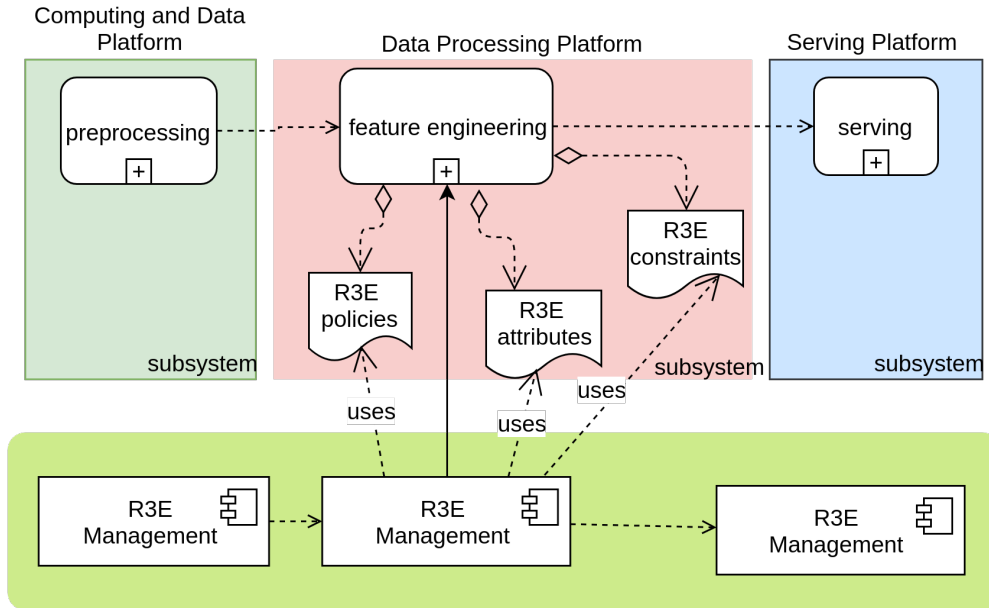


Figure 6: R3E Reactive systems with group of management

Distributed controls: Each component, based on the view in Figure 1, can be controlled and managed through the individual component's R3E object. For example, data collections can be controlled to select suitable data sources and such controls are independent from another control of the ML model service. However, from an end-to-end viewpoint, we need to coordinate these controls to achieve the defined QoAChain for the whole *BDML* system.

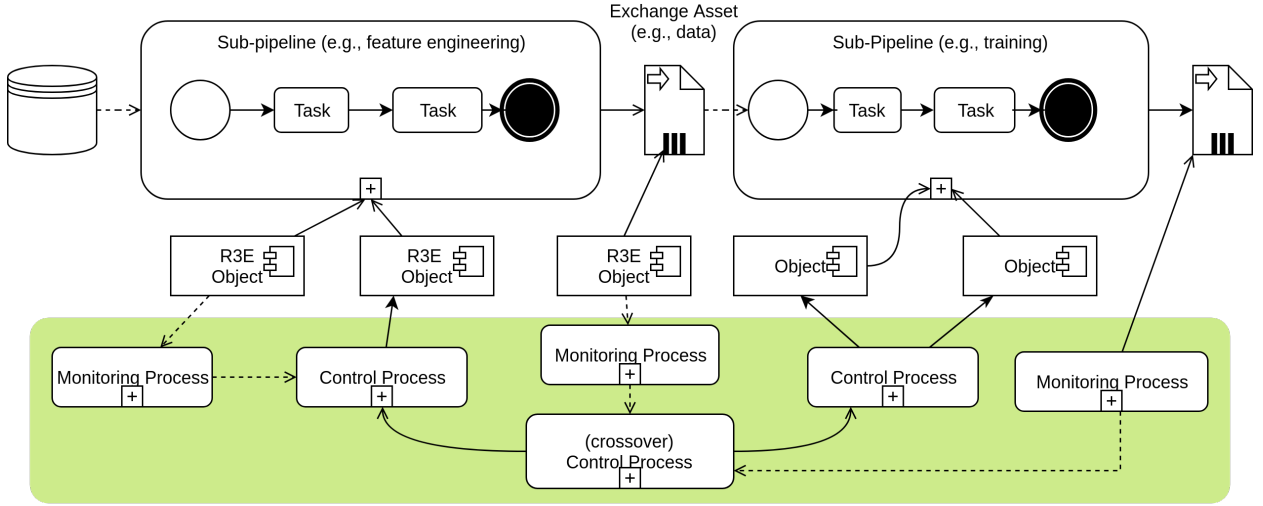


Figure 7: Distributed monitoring and control processes utilizing R3E objects

Figure 7 presents our approach to control ML tasks and corresponding components by using Monitoring Process and Control Process to interact with R3EObject. For each subsystem and sub-pipelines on that subsystem, R3EObjects are used to monitor and control tasks and components. Corresponding Monitoring Process and Control Process will interact with these R3EObject. Exchanges among subpipelines, such as the featuring data outputted from feature engineering subpipeline to training subpipeline will be also monitored. Individual Control Process for different subpipelines will be coordinated by a crossover Control Process. Thus, we have distributed controls but through a centralized coordination. In our implementation, we plan to implement functions in Monitoring Process and Control Process using serverless frameworks. Note that Monitoring Process will need to work with monitoring systems that we will discuss in the next section.

3.3.2. Monitoring and Analytics

Monitoring and analytics monitor and analyze R3E attributes of services, data and ML models and map the R3E attributes to R3E objects. For the whole approach, we need to leverage different methods for monitoring and analytics. Shown in Figure 8, we will need (i) monitoring probes and frameworks, (ii) tests and benchmarks units and frameworks, and (iii) data analysis/machine learning for understanding monitoring data. We will support different scopes of monitoring and analytics: the system as a whole, subsystem/subpipeline and component/task. With such scopes, we will focus on end-to-end aspects. For example, data reliability can be examined along the path from data sources to the final inference results. The consumer can also expect an end-to-end R3E attribute, such as accuracy and response time, which can only be achieved if we are able to monitor different parts and to perform coordination-aware assurance, e.g., using elasticity principles, at the system as a whole scope. For the implementation, we will need to integrate various monitoring systems for services (such as Prometheus), for data (e.g., data validation tools from `scikit-learn` and TensorFlow Data Validation), and for ML models (e.g., extracted from ML frameworks).

In terms of analytics, we also have different perspectives about which techniques can be applied for which parts and whether we can have evaluation and interpretation in a subjective manner. Here the context of analytics is important. Therefore, we suggest to use a context model of What, When, Where, Who and How for analyzing R3E. Figure 9 presents the context model for understanding R3E attributes. Our approach in the implementation is to extend the work on monitoring of ML service contracts (Truong and Nguyen, 2021) to cover R3E.

3.3.3. Testing, Benchmarking, and Experimenting for R3E

To run tests, benchmarks and experiments (TBE) is of paramount importance for optimizing R3E. The challenge is that it is not just related to ML training and serving, but also to other tasks in the whole *BDML* system. Testing, benchmarking and experimenting have to do across subsystems and focus on correlating R3E issues. Current ML testing frameworks are mainly focused on ML models (Aggarwal et al., 2019; Riccio et al., 2020). Testing big data is currently focused on data storage and querying (Alexandrov et al., 2013; Li et al., 2016; Baru et al., 2012; Gulzar et al.,

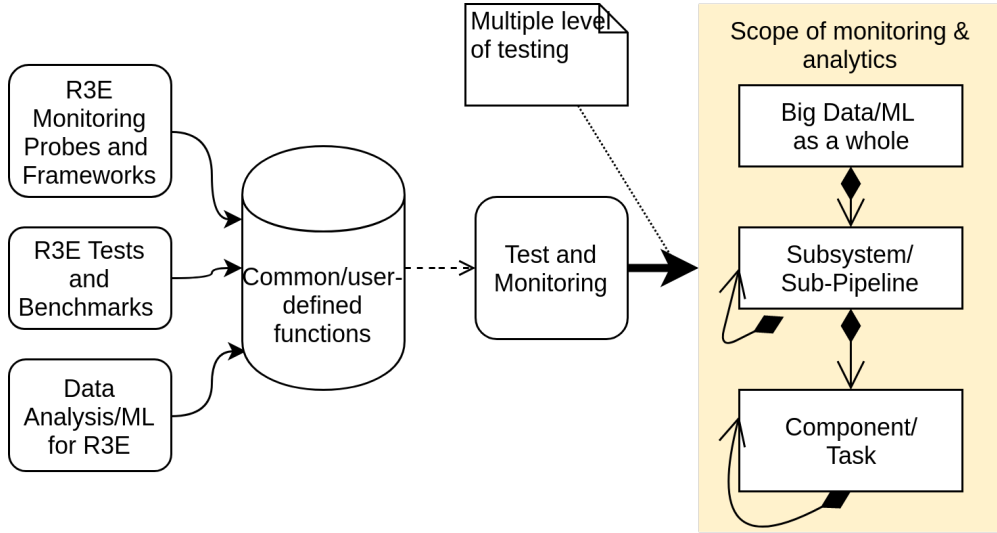


Figure 8: Scopes of tests, benchmarks and experiments

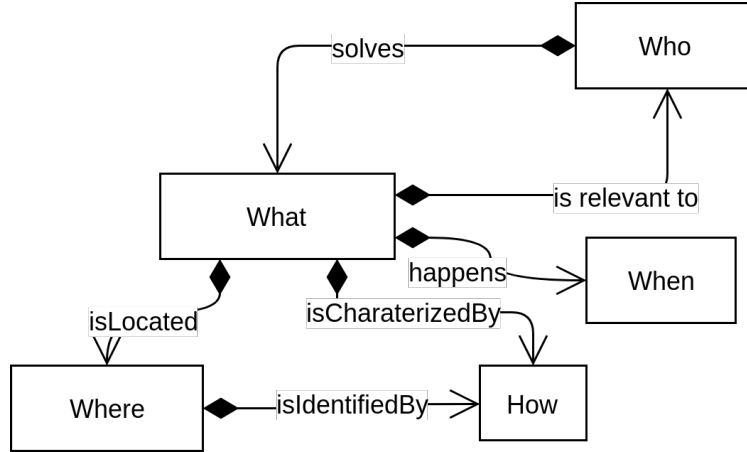


Figure 9: R3E W4H analytics context

2019; Bajaber et al., 2020). Combination of different tests into a coherent R3E view is currently missing. Furthermore, ML experiment solutions (Gharibi et al., 2019; Duarte et al., 2017) focus on mainly model metadata, used data sets, and hyperparameters but the management of data sources, services performances and code/data versions is not integrated.

Figure 10 outlines our approach for testing, benchmarking, and experimenting. R3E TBEUnit is an abstract unit designed for testing, benchmarking, and experimenting. At the top level, we use workflows to coordinate tests/benchmarks across subsystems for different subpipelines. We will develop a variety of R3E test/benchmark units for *BDML*; each unit does test/benchmark not only one component but also a layer or an aspect, e.g., data. Last, R3E integration tests/benchmarks/experiments carried out for individual subsystems and components are linked together.

4. Illustrative Examples

In this section, we explain strategies for optimizing the BIM scenario mentioned in Section 2.2. Considering that we must have to apply the R3E approach for the BIM scenario to allow the optimization of the BIM ML pipeline in an end-to-end manner. To this end, we analyze the scenario and apply step-by-step of the R3E approach mentioned in

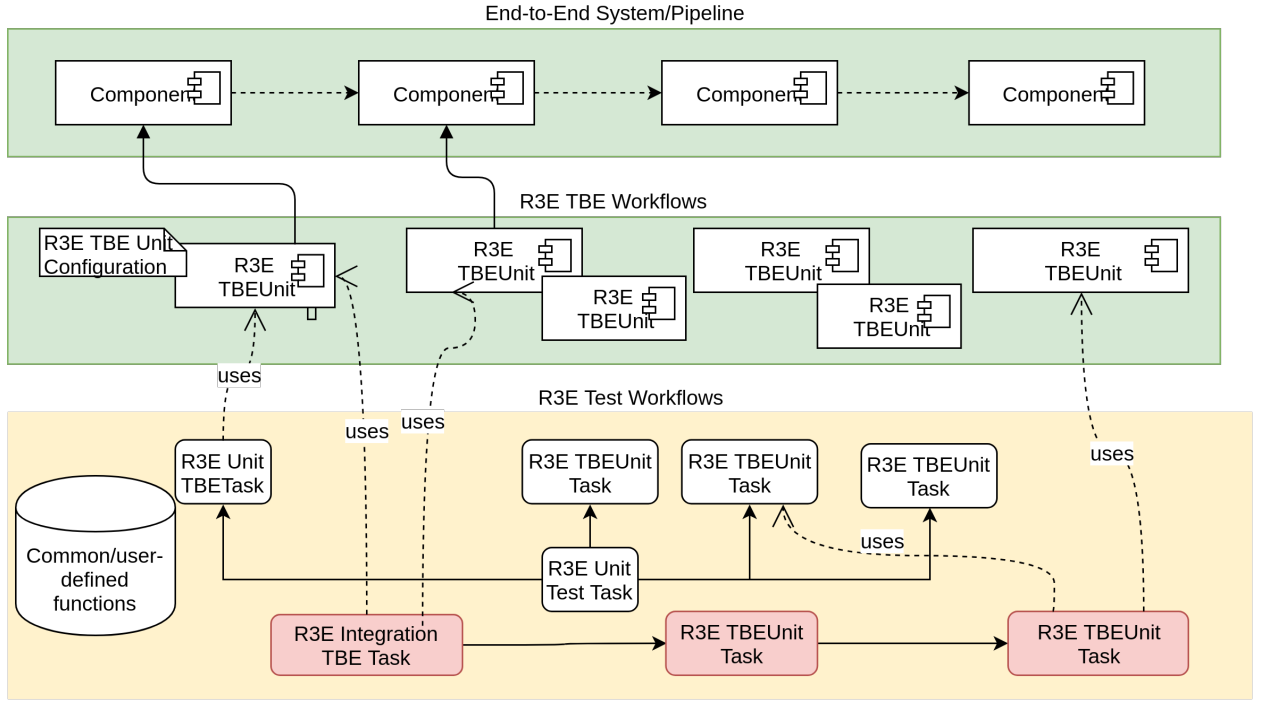


Figure 10: Conceptual view of R3E tests, benchmarks and experiments

Section 3. In the following we summarize R3E aspects, identified requirements to be addressed. For each category, we only show key examples.

In terms of **QoAChain** we have identified:

Aspects	Identified requirements
model accuracy	as a contractual means between BIM and customers for the whole system
response time	as a contractual means between the ML service provider and customers for inferencing
accuracy & response time tradeoffs	accuracy is more important than response time to customers. Therefore, elasticity of cloud resources for ML services can be flexible (using CPU, GPU and even spot instances)

The accuracy and response time tradeoffs are based on the business of the BIM scenario: it is important to predict and classify building objects with a high degree of accuracy to make sure that the design is correct and reliable. For this, the customer does not need a real time prediction. Consequently, in terms of R3E at runtime, computing and data resources could be allocated differently with respect to the cost. For example, GPU resources might be used only if a higher cost is accepted by the customer, whereas more computing resources are needed for data preprocessing and feature engineering phases.

In terms of **R3E objects and operators**, we have identified:

Aspects	Identified requirements
R3E objects	include (i) data resources collector and selector (trustful data sources, text and 3D data); (ii) feature engineering component (3D data extraction granularity); (iii) machine learning models (model versions and parameters); (iv) ML services (coupled with ML models and underlying computing resources); computing resources (cloud-based CPU & GPU resources)
Operators	data feature engineering operators are for fine-grained and high-grained of data extraction; changes of ML models and parameters; elastic computing resources with possibility to have different types of resources, including edge hardware, cloud-based CPU and GPU

The data resource collector and selector are not designed as data sources are typically fixed, e.g., from S3 storage or shared file-based storage. This requires the design of new components for data collector and selector that are integrated with data source metadata and data resource catalogs as well as the change of the data pipeline from User Tool to Data Service (see Figure 2). One example is to use DVC¹ in combination with quality of data metrics, such as trust, data completeness, and timeliness, that are determined during the data export task. Another aspect is to control feature engineering task which is tightly coupled with preprocessing but strongly influences the accuracy, cost and time of the prediction in ML models. This requires us to separate *preprocessing* and *feature engineering* tasks.

In terms of **R3E Engineering**, we have identified:

Aspects	Identified requirements
Coordination for R3E	three subsystems are identified: preprocessing, featuring engineering, and serving. They can be optimized independently or together
Monitoring and Analytics	quality of data in data collection; model accuracy metrics during serving; performance response time; costs paid to cloud resources
Tests, benchmarks and experiments	tests of data validation; benchmarks in training; monitoring and provenance for data resources and machines for individual experiments

Through the separation between *preprocessing* and *feature engineering*, and introduce an R3E object between the two tasks for controlling feature engineering. Finally, ML models need to be controlled separately through QoAChain coupling the ML models with resource elasticity.

5. Discussion

Different communities have advocated R3E in different ways. In ML benchmarks, various initiatives for testing robustness, performance and cost have been carried out (Fei Tang et. al, 2020). Recently the discussion of end-to-end ML has been attracted many researchers. Especially, in the software engineering and ML production, the optimization of many phases in end-to-end ML pipelines is on the focus (Amershi et al., 2019). AIOps (Dang et al., 2019) focuses on using AI to optimize quality of software. Our work has a similar ultimate goal but our approach differs as we focus on software service programming, engineering and analytics. To our best knowledge there is no previous roadmap for R3E for end-to-end *BDML*.

Different components of an end-to-end *BDML* system have different R3E attributes and concerns that lead to different optimization focuses of R3E. For example, in ML services and models, robustness as a critical concern has been discussed intensively (Sehwag et al., 2019). However, the elasticity of ML services has not been studied well (Huang et al., 2015), while the elasticity of infrastructural resources for enabling cloud computing has been studied intensively. Another example is that, while data is very important for robustness in ML training and ML services, monitoring quality of data monitoring and supporting data resources elasticity in ML have not been well developed. The reliability, reflecting the concept of offering “reliable service” (Kumar and Vidhyalakshmi, 2018; Galetzka et al., 2006), for different individual components (e.g., ML services, data stores, and data brokers) has been studied intensive but the reliability of *BDML* systems has not been studied well in an end-to-end manner. Resilience (Robbins et al., 2012) is mostly addressed at the system services. We need to understand how ML models resilience (Park et al., 2017) is related to elasticity of data and other aspects, e.g., message middleware (Wang et al., 2010) and programming languages (Grove et al., 2019) in our pipeline design.

6. Conclusions and Future Work

In this chapter we present a novel conceptual approach for implementing robustness, reliability, resilience, and elasticity for end-to-end big data machine learning systems, called R3E. Given R3E attributes, we proposed to use QoAChain as a contractual means for specifying constraints of R3E. To manage R3E of components and tasks, from the R3E view, we abstract them under R3E objects and devise operations for monitoring, controlling and optimizing R3E. Our approach has presented key engineering methods for main design and engineering activities with respect to R3E. In our current work, we have not addressed all tools implemented our abstractions and engineering methods. We are working on the monitoring and observability of ML services (Truong and Nguyen, 2021) and a service for collecting trails from tests, benchmarks, and experiments for end-to-end ML systems. However, this chapter layouts fundamental steps for addressing R3E design and engineering in the future.

¹<https://dvc.org>

We foresee that different scenarios can be elaborated to have a deeper view on R3E. Details of tools and components can be carried out for training optimization, runtime ML model serving, out-of-distribution detection and optimization, and elasticity serving. The situation is even more challenging when *BDML* systems have more distributed learning (Verbraeken et al., 2020), as the nature of complexity is increasing. We will revise our approach to address such new development. Our current work is to focus on two aspects: end-to-end self-optimized solutions and QoAChain toolset.

Acknowledgments

We are grateful to Minjung Ryu for the discussion about the case of ML for classification of building information modeling (BIM) elements.

References

- Ackley, D.H., 2013. Beyond efficiency. *Commun. ACM* 56, 38–40. doi:10.1145/2505340.
- Aggarwal, A., Lohia, P., Nagar, S., Dey, K., Saha, D., 2019. Black box fairness testing of machine learning models, in: Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Association for Computing Machinery, New York, NY, USA. pp. 625–635. doi:10.1145/3338906.3338937.
- Alexandrov, A., Brücke, C., Markl, V., 2013. Issues in big data testing and benchmarking, in: Proceedings of the Sixth International Workshop on Testing Database Systems, Association for Computing Machinery, New York, NY, USA.
- Amershi, S., Begel, A., Bird, C., DeLine, R., Gall, H., Kamar, E., Nagappan, N., Nushi, B., Zimmermann, T., 2019. Software engineering for machine learning: A case study, in: Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice, IEEE Press. pp. 291–300.
- Bajaber, F., Sakr, S., Batarfi, O., Altalhi, A., Barnawi, A., 2020. Benchmarking big data systems: A survey. *Computer Communications* 149, 241 – 251.
- Balint, F., Truong, H.L., 2017. On supporting contract-aware iot dataspace services, in: 5th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, MobileCloud 2017, San Francisco, CA, USA, April 6-8, 2017, IEEE Computer Society. pp. 117–124. URL: <https://doi.org/10.1109/MobileCloud.2017.28>, doi:10.1109/MobileCloud.2017.28.
- Baru, C., Bhandarkar, M., Nambiar, R., Poess, M., Rabl, T., 2012. Big data benchmarking, in: Proceedings of the 2012 Workshop on Management of Big Data Systems, Association for Computing Machinery, New York, NY, USA. pp. 39–40.
- Blanchard, P., Mhamdi, E.M.E., Guerraoui, R., Stainer, J., 2017. Machine learning with adversaries: Byzantine tolerant gradient descent, in: Guyon, I., von Luxburg, U., Bengio, S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., Garnett, R. (Eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, 4-9 December 2017, Long Beach, CA, USA, pp. 119–129.
- Brtis, J., Jackson, S., Cureton, K., 2021. https://www.sebokwiki.org/wiki/System_Resilience. Last access: October 15, 2021.
- Dang, Y., Lin, Q., Huang, P., 2019. Aiops: Real-world challenges and research innovations, in: Proceedings of the 41st International Conference on Software Engineering: Companion Proceedings, IEEE Press. pp. 4–5.
- Duarte, J.C., Cavalcanti, M.C.R., de Souza Costa, I., Esteves, D., 2017. An interoperable service for the provenance of machine learning experiments, in: Proceedings of the International Conference on Web Intelligence, Association for Computing Machinery, New York, NY, USA. p. 132–138.
- Dustdar, S., Guo, Y., Satzger, B., Truong, H.L., 2011. Principles of elastic processes. *IEEE Internet Comput.* 15, 66–71.
- Elsayed, E.A., 2012. *Reliability Engineering*. 2nd ed., Wiley Publishing.
- Fei Tang et. al, 2020. AIBench: An Industry Standard AI Benchmark Suite from Internet Services. Technical Report. BenchCouncil: International Open Benchmarking Council.
- Fischer, L., Memmen, J., Veith, E.M.S.P., Tröschel, M., 2018. Adversarial resilience learning - towards systemic vulnerability analysis for large and complex systems. *CoRR abs/1811.06447*. arXiv:1811.06447.
- Galetzka, M., Verhoeven, J., Pruyn, A., 2006. Service validity and service reliability of search, experience and credence services: A scenario study. *International journal of service industry management* 17, 271–283.
- Gharibi, G., Walunj, V., Rella, S., Lee, Y., 2019. Modelkb: Towards automated management of the modeling lifecycle in deep learning, in: Proceedings of the 7th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering, IEEE Press. pp. 28–34.
- Gribble, S.D., 2001. Robustness in complex systems, in: Proceedings Eighth Workshop on Hot Topics in Operating Systems, pp. 21–26.
- Grove, D., Hamouda, S.S., Herta, B., Iyengar, A., Kawachiya, K., Milthorpe, J., Saraswat, V., Shinnar, A., Takeuchi, M., Tardieu, O., 2019. Failure recovery in resilient x10. *ACM Trans. Program. Lang. Syst.* 41.
- Gujarati, A., Elnikety, S., He, Y., McKinley, K.S., Brandenburg, B.B., 2017. Swayam: Distributed autoscaling to meet slas of machine learning inference services with resource efficiency, in: Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference, Association for Computing Machinery, New York, NY, USA. pp. 109–120.
- Gulzar, M.A., Mardani, S., Musuvathi, M., Kim, M., 2019. White-box testing of big data analytics with complex user-defined functions, in: Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Association for Computing Machinery, New York, NY, USA. pp. 290–301.
- Harlap, A., Tumanov, A., Chung, A., Ganger, G.R., Gibbons, P.B., 2017. Proteus: Agile ml elasticity through tiered reliability in dynamic resource markets, in: Proceedings of the Twelfth European Conference on Computer Systems, Association for Computing Machinery, New York, NY, USA. pp. 589–604.
- Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., Meger, D., 2017. Deep reinforcement learning that matters, in: Thirtieth AAAI Conference On Artificial Intelligence (AAAI), 2018.

- Hendrycks, D., Dietterich, T.G., 2019. Benchmarking neural network robustness to common corruptions and perturbations. CoRR abs/1903.12261. [arXiv:1903.12261](#).
- Huang, B., Boehm, M., Tian, Y., Reinwald, B., Tatikonda, S., Reiss, F.R., 2015. Resource elasticity for large-scale machine learning, in: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Association for Computing Machinery, New York, NY, USA. pp. 137–152.
- Katzir, Z., Elovici, Y., 2018. Quantifying the resilience of machine learning classifiers used for cyber security. Expert Systems with Applications 92, 419–429.
- Kleppmann, M., 2016. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. O'Reilly.
- Kulkarni, A., Chong, D., Batarese, F.A., 2020. 5 - foundations of data imbalance and solutions for a data democracy, in: Batarese, F.A., Yang, R. (Eds.), Data Democracy. Academic Press, pp. 83–106. URL: <https://www.sciencedirect.com/science/article/pii/B9780128183663000058>, doi:<https://doi.org/10.1016/B978-0-12-818366-3.00005-8>.
- Kumar, V., Vidhyalakshmi, R., 2018. Reliability Aspect of Cloud Computing Environment. 1st ed., Springer Publishing Company, Incorporated.
- Laranjeiro, N., Agnello, J.a., Bernardino, J., 2021. A systematic review on software robustness assessment. ACM Comput. Surv. 54. URL: <https://doi.org/10.1145/3448977>, doi:10.1145/3448977.
- Lee, D., 2019. Big data quality assurance through data traceability: A case study of the national standard reference data program of korea. IEEE Access 7, 36294–36299.
- Li, N., Lei, Y., Khan, H.R., Liu, J., Guo, Y., 2016. Applying combinatorial test data generation to big data applications, in: Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering, Association for Computing Machinery, New York, NY, USA. pp. 637–647.
- Littlewood, B., Strigini, L., 2000. Software reliability and dependability: A roadmap, in: Proceedings of the Conference on The Future of Software Engineering, Association for Computing Machinery, New York, NY, USA. pp. 175–188.
- Nguyen, C., Mehta, A., Klein, C., Elmroth, E., 2019. Why cloud applications are not ready for the edge (yet), in: Proceedings of the 4th ACM/IEEE Symposium on Edge Computing, Association for Computing Machinery, New York, NY, USA. pp. 250–263. doi:10.1145/3318216.3363298.
- Park, S., Weimer, J., Lee, I., 2017. Resilient linear classification: An approach to deal with attacks on training data, in: Proceedings of the 8th International Conference on Cyber-Physical Systems, Association for Computing Machinery, New York, NY, USA. p. 155–164.
- Ricco, V., Jahangirova, G., Stocco, A., Humbatova, N., Weiss, M., Tonella, P., 2020. Testing machine learning based systems: a systematic mapping. Empir. Softw. Eng. 25, 5193–5254. URL: <https://doi.org/10.1007/s10664-020-09881-0>, doi:10.1007/s10664-020-09881-0.
- Robbins, J., Krishnan, K., Allspaw, J., Limoncelli, T., 2012. Resilience engineering: learning to embrace failure. Commun. ACM 55, 40–47. URL: <https://doi.org/10.1145/2366316.2366331>, doi:10.1145/2366316.2366331.
- Ryu, M., Truong, H.L., Kannala, M., 2021. Understanding quality of analytics trade-offs in an end-to-end machine learning-based classification system for building information modeling. J. Big Data 8, 31. URL: <https://doi.org/10.1186/s40537-021-00417-x>, doi:10.1186/s40537-021-00417-x.
- Saria, S., Subbaswamy, A., 2019. Tutorial: Safe and reliable machine learning. CoRR abs/1904.07204. [arXiv:1904.07204](#).
- Sehwag, V., Bhagoji, A.N., Song, L., Sitawarin, C., Cullina, D., Chiang, M., Mittal, P., 2019. Analyzing the robustness of open-world machine learning, in: Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security, Association for Computing Machinery, New York, NY, USA. pp. 105–116. doi:10.1145/3338501.3357372.
- Smith, J., 2018. Machine Learning Systems: Designs That Scale. 1st ed., Manning Publications Co., USA.
- Suryavansh, S., Bothra, C., Chiang, M., Peng, C., Bagchi, S., 2019. Tango of edge and cloud execution for reliability, in: Proceedings of the 4th Workshop on Middleware for Edge Clouds & Cloudlets, Association for Computing Machinery, New York, NY, USA. p. 10–15. doi:10.1145/3366614.3368103.
- Trivedi, K.S., Kim, D.S., Ghosh, R., 2009. Resilience in computer systems and networks, in: 2009 IEEE/ACM International Conference on Computer-Aided Design - Digest of Technical Papers, pp. 74–77.
- Truong, H.L., Comerio, M., Paoli, F.D., Gangadharan, G.R., Dustdar, S., 2012. Data contracts for cloud-based data marketplaces. Int. J. Comput. Sci. Eng. 7, 280–295. URL: <https://doi.org/10.1504/IJCSE.2012.049749>, doi:10.1504/IJCSE.2012.049749.
- Truong, H.L., Murguzur, A., Yang, E., 2018. Challenges in enabling quality of analytics in the cloud. J. Data and Information Quality 9. doi:10.1145/3138806.
- Truong, H.L., Nguyen, T., 2021. Qoa4ml – a framework for supporting contracts in machine learning services, in: 2021 IEEE International Conference on Web Services (ICWS), IEEE, United States.
- Verbraeken, J., Wolting, M., Katzy, J., Kloppenburg, J., Verbelen, T., Rellermeyer, J.S., 2020. A survey on distributed machine learning. ACM Comput. Surv. 53.
- Wang, J., Balazinska, M., 2017. Elastic memory management for cloud data analytics, in: Silva, D.D., Ford, B. (Eds.), 2017 USENIX Annual Technical Conference, USENIX ATC 2017, Santa Clara, CA, USA, July 12–14, 2017, USENIX Association. pp. 745–758.
- Wang, J., Jiang, P., Bigham, J., Chew, B., Novkovic, M., Dattani, I., 2010. Adding resilience to message oriented middleware, in: Proceedings of the 2nd International Workshop on Software Engineering for Resilient Systems, Association for Computing Machinery, New York, NY, USA. p. 89–94.
- Yang, F., Chien, A.A., Gunawi, H.S., 2017. Resilient cloud in dynamic resource environments, in: Proceedings of the 2017 Symposium on Cloud Computing, Association for Computing Machinery, New York, NY, USA. p. 627.
- Zhang, A., Song, S., Wang, J., Yu, P.S., 2017. Time series data cleaning: From anomaly detection to anomaly repairing. Proc. VLDB Endow. 10, 1046–1057.
- Zhang, P., Cao, W., Muccini, H., 2020. Quality assurance technologies of big data applications: A systematic literature review. CoRR abs/2002.01759. [arXiv:2002.01759](#).
- Zheng, Y., Xu, L., Wang, W., Zhou, W., Ding, Y., 2017. A reliability benchmark for big data systems on jointcloud, in: 2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW), pp. 306–310.

7. Biography

Hong-Linh Truong is an associate professor (tenured) at Department of Computer Science, School of Science, Aalto University, Finland. He leads the AaltoSEA Group on Systems and Services Engineering Analytics. He received his habilitation in 2013 and PhD in 2005 from TU Wien, Austria. His main research interest focuses Systems, Software, Data and Service Engineering Analytics by developing novel techniques and tools for monitoring, analyzing, and optimizing functions, performance, data quality, elasticity, and uncertainties associated with systems, software, data and services.