Modelling Data Pipelines

Modelling Data Pipelines

Aiswarya Raj, Jan Bosch
Chalmers University of Technology
Göteborg, Sweden
{*aiswarya,jan.bosch*}*@chalmers.se*

Helena Holmström Olsson
Malmö University
Malmö, Sweden
helena.holmstrom.olsson@mau.se

Tian J. Wang
Ericsson
Göteborg, Sweden
tian.j.wang@ericsson.com

***Abstract*—Data is the new currency and key to success. However, collecting high-quality data from multiple distributed sources requires much effort. In addition, there are several other challenges involved while transporting data from its source to the destination. Data pipelines are implemented in order to increase the overall efficiency of data-flow from the source to the destination since it is automated and reduces the human involvement which is required otherwise.**

Despite existing research on ETL (Extract-Transform-Load) and ELT (Extract-Load-Transform) pipelines, the research on this topic is limited. ETL/ELT pipelines are abstract representations of the end-to-end data pipelines. To utilize the full potential of the data pipeline, we should understand the activities in it and how they are connected in an end-to-end data pipeline. This study gives an overview of how to design a conceptual model of data pipeline which can be further used as a language of communication between different data teams. Furthermore, it can be used for automation of monitoring, fault detection, mitigation and alarming at different steps of data pipeline.

***Index Terms*—Data pipelines; conceptual model; data workflow; domain specific language; agile methodology**

I. Introduction

Data is becoming increasingly popular in the industry due to the importance of data products such as APIs, dashboards, benchmarks and report creations. The role data plays in the decision-making process and the development of ML and DL models makes it even more important. Therefore, all processes associated with data ranging from data generation to data reception need to be monitored. Fault detection, reporting and mitigating the effect of faults are complex but inevitable while building efficient data products.

Data pipelines are complex chains of activities that manipulate data where the output of one component becomes the input to the other [1] thereby allowing smooth, automated flow of data from source to destination. A data pipeline starts with a data source that generates data and ends at a destination that receives the processed data. The ultimate destination of a data pipeline need not be data storage. Instead it can be any application such as a visualization tool [2] [3], Machine Learning(ML) models [4] [5] or Deep Learning(DL) models [6] [7]. Components in the data pipeline are capable of automating processes involved in extracting, transforming, combining, validating, and loading data [8]. Data pipelines can process different types of data such as continuous, intermittent and batch data [9]. Moreover, data pipelines eliminate errors and accelerate the end-to-end data processes which in turn reduces the latency in the development of data products. Hence, the usage of data pipelines is an absolute necessity for all data-driven companies.

Although data pipelines have the potential to overcome data management challenges through automation, monitoring, fault detection, etc, modeling data pipelines for a use case demands an unreasonable amount of time and effort. We need to identify the activities which consume data, the output of each activity, order of execution, monitoring methods, intermediate storages, where to place the storage in the pipeline, data collection method, etc and they varies between companies. Thus, modeling a data pipeline is important as well as time-consuming.

This study addresses the above mentioned problems by proposing a conceptual model which is developed based on a multiple case study performed at a large-scale telecommunication company. The contribution of this paper is three-fold. First, it describes the challenges associated with data management and the usage of existing data pipelines. Second, a conceptual model of an end-to-end data pipeline is presented that can be used as a reference while building data pipelines for applications such as ML/DL models to incorporate automatic monitoring, fault detection, mitigation and alarming techniques. The conceptual model is validated through a case study with three leading companies from manufacturing, telecommunication and automobile domains. Furthermore, the paper maps the challenges that can be potentially solved by the usage of the proposed data pipeline model.

The remainder of this paper is organized as follows. In the next section, we present the background of the study. Section III discusses the research methodology adopted for conducting the study. Section IV introduces the use cases and section V describes the challenges using the existing pipelines. Section VI details the data pipeline meta-model. Section VII describes the conceptual model that we use as a basis for our analysis. A validation study is detailed in section VIII and section IX outlines the threats to validity. Section X summarizes our study and the conclusions.

II. Background

Data-driven development has been adopted by the companies realizing the benefits that can be yield from data [10]. Technologies for data-driven development needs enormous amount of data for their processing. Consequently, collection, storage, and processing of copious amounts of data became a necessity [11]. With the increased amount of data, data

978-1-7281-9532-2/20/$31.00 ©2020 IEEE
DOI 10.1109/SEAA51224.2020.00014

management challenges also increased [12]. Data pipelines can be a potential solution to overcome at least some of these challenges.

Data pipelines are broadly classified into two categories namely ETL and ELT. ETL stands for Extract, Transform and Load whereas ELT stands for Extract Load and Transform. P. Vassiliadis has presented a survey on Extraction-Transformation-Loading (ETL) processes and tools. The study describes a standardized approach for the construction of conceptual and logical modeling tools for ETL processes [13]. Also, each component of the E-T-L triplet is analyzed separately for the activities happening inside each component, identified the real-time challenges associated with each of them and solutions to overcome those challenges are explained in detail.

In [14], the authors have proposed a UML based conceptual approach to model ETL processes. A group of UML concepts is utilized to represent the ETL processes such as data sources integration, transformation, key generation and so on. The authors claim that a UML based approach makes the model simple to understand and powerful. Tilmann and Hans have described a big data analytics pipeline with abstract stages of it [15]. They also discuss the necessity of ETL-like processes in benchmarking and has proposed and implemented a framework similar to ETL.

A machine learning pipeline is proposed by Amershi et. al in [16] based on a case study at Microsoft. It discusses the nine stages of ML workflow along with the best practices followed at Microsoft. The authors mention the importance of data in AI applications by illustrating the three data related stages in ML workflow.

Although these studies lay strong foundation, practitioners experience several data quality challenges and issues around data governance and security while dealing with real-time data. Our study aims to design a conceptual model that can act as a domain specific language for fault tolerant data pipelines.

III. Research Methodology

The objective of this study is to understand the existing data pipeline as well as the challenges experienced at the case company and to develop a conceptual model of the robust data pipeline. Based on the study objectives, we formulated the following research questions:

- **RQ1:** What are the challenges related to data and data pipeline management that practitioners in the case company experience?
- **RQ2:** What are the essential elements of a fault-tolerant, automated, traceable end-to-end data pipeline?

The research methodology adopted for the study is illustrated in fig. 1.

A. Exploratory Case Study

A qualitative approach was chosen for the case study as it allows the researchers to explore, study and understand the real-world cases in its context in more depth [17]. Since the concept of the data pipeline is a less explored topic in research, we have adopted a case study approach [18]. Each case in the study pertains to a use case that makes use of data. Although the cases in our study are different use cases from the same company, they utilize data for different activities and can be benefited from the data pipeline we develop. Three selected use-cases at the company are given in table 1.

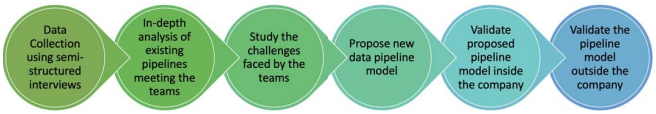

Fig. 1. Research Methodology

TABLE I
DESCRIPTION OF USE CASES AND ROLES OF THE INTERVIEWEES

Case ID	Use cases at case company	Interviewed Experts	
		ID	Role
A	Data collection pipeline for data analytics	R1	Senior Data Scientist
		R2	Data Scientist
B	Building data pipelines for data governance	R3	Data Scientist
		R4	Analytics System Architect
		R5	Software Developer
C	Machine learning pipeline	R6	Data Scientist
		R7	Senior Data Scientist
		R8	Software Developer
		R9	Senior Data Scientist

B. Data Collection

Qualitative data was collected by means of interviews and meetings [19]. Based on the objective of the research, to explore and study the applications consuming data in the company, an interview guide with 45 questions categorized into six sections was formulated. The first and second sections focused on the background of the interviewee. The third and fourth sections focused on the data collection and processing in various use-cases and the last section inquired about data testing and monitoring practices and the impediments faced during each step of the data pipeline. The interview guide was prepared by the first author and was reviewed by second and third authors. According to the recommendations, extra questions were added, a few similar and irrelevant questions were removed and some questions were modified. Finally, an interview protocol with 30 questions across six different categories was developed. All except three interviews were conducted via videoconferencing. Each interview lasted 50 to 100 minutes. The interviews were recorded with the permission of respondents and were transcribed later for analysis. One of the authors works in the case company and the first author is a consultant who attends weekly meetings with data scientists and data analysts. Data collected through the meetings and discussions are also incorporated.

C. Data Analysis

The audio recordings of interviews were transcribed and a summary of it was prepared by the first author. The transcripts were investigated for relations, similarities, and dissimilarities. The interview transcripts were open coded following the

guidelines in [20]. The first author prepared notes during the meetings with the team and analyzed them further. The main contact point who is also an author helped with analyzing the parts of the pipeline as well as the infrastructure used for building that. These notes together with the codes from transcripts were further analyzed to obtain an end-to-end view of different use cases. It also helped to understand the parts common to all use cases. After careful analysis of collected data and based on the inputs from the other two authors, the first author developed the first conceptual model which got refined through iterations.

D. Validation Study

The validation study is performed both internally and externally through qualitative interviews followed by feedback sessions. First, the conceptual model of the data pipeline was presented by the first author to the teams inside the case company. The reflections about the data pipeline, overall opinion, agreements and suggestions for improvement were collected from every practitioner. These reflections are considered as internal validation.

For external validation of our findings, two manufacturing companies were selected. An interview guide was prepared by the first author for validating the conceptual data pipeline model. The first author presented the model. Second and third authors conducted an interview followed by a discussion to collect data. The entire session was recorded for the first case and for the second one, the first author took notes. Thus, feedback from 20 practitioners was collected and recorded. The conceptual model was then modified to address some of the issues raised during the discussions. Also, the inputs from the practitioners are incorporated in the conceptual model. The remaining issues will be addressed in future works.

IV. Use cases

This section introduces the existing data pipelines used in the telecommunication firm. Each of these use cases is separate and there is no interaction between those pipelines.

A. Data Collection Process

The company collects data from multiple sources distributed across the globe which is a challenging activity. When data is collected from a device located in another country or from the customer network, it should be according to the legal agreement. Also, sensitive information in the data should be handled responsibly. Furthermore, data collection should consider the fact that different data sources generate data in different frequencies and formats. For instance, data can be collected continuously, intermittently or as batches. Moreover, the data collection mechanism itself should be capable to adjust with different intensities of data-flow.

When data collection is automated, these challenges should be addressed properly. Fig. 2 shows the automatic data collection from the devices. In this scenario, the device is placed inside a piece of equipment owned by customers. However, the device data is extracted excluding the customer's sensitive

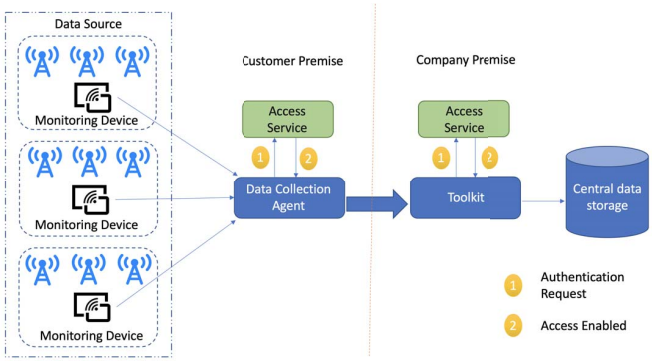

Fig. 2. Data Collection Process

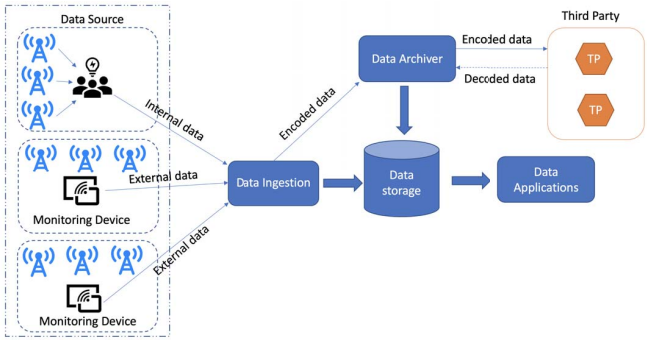

Fig. 3. Data Pipeline for Data Governance

information. Base stations have got nodes as well as a device for monitoring and managing the nodes. Data collection agents are equipment located on the customer's premises (physical location) that can interact either with the nodes directly or with the device to collect the data. However, to ensure that the data collection agent has the right to collect data, it is authenticated with the help of access service. The data thus collected is transmitted through a secure tunnel to the toolkit located at the company premise. This data collection agent also needs the help of access service for authentication. Once the agent at the customer premise gets the data, it is stored in the central data storage. The teams can get the data from the central data storage.

B. Pipeline for Data Governance

The data pipeline shown in fig. 3 is developed to serve the teams in the company who are working with data whenever they need it (With the term 'data', we mean the link from which the original data can be downloaded). This data pipeline gets two types of data dumps: internal and external. The internal data dump is the data that is ingested by the teams inside the company and external data dump is the data collected directly from the devices in the fields. The data ingestion method is different for different sources and the ingested data is stored in the data storage for further use. The data can have encrypted links that need to be decrypted before storing it. Whenever an encrypted data dump is found, the data archiver

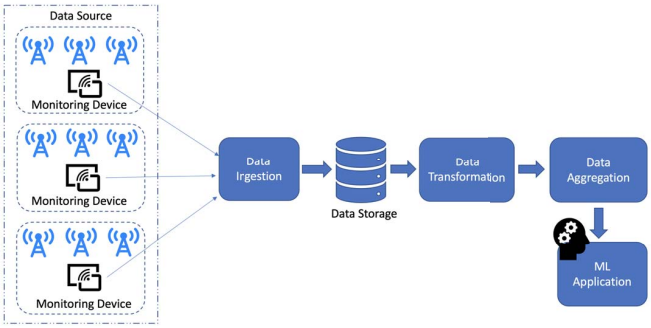

Fig. 4. Pipeline for Machine Learning Systems

module sends it to third-party services for decryption. Decoded links from the third party are stored. Thus data from different sources are made available in a central location. Teams can request data from any stage of the pipeline. The pipeline is manually monitored by 'flow guardian' who is responsible for identifying the faults and solving them.

C. Pipeline for Machine Learning Systems

Data pipeline has four main steps namely ingest, store, transform and aggregate. Data is generated by the source which is gathered by a special zone in the field. The data ingestion module is connected to those zones in the field and the collected data is ingested into the pipeline as batches. When new compressed files are found during periodic checks, the transaction is logged and downloads it. These new files are then loaded into the archive directory of the data cluster. The data stored in the cluster cannot be used directly by the ML applications. Moreover, the data logs collected from different devices can be of different formats. They need to be converted to a suitable format. This conversion is performed by the data transformation module. Data transformation checks for the new files in the archive directory of the data cluster and when found, it is fetched, uncompressed and processed to convert it to an appropriate format. The converted data is then given as input to the data aggregation module where the data is aggregated and summarized to form structured data which is further given as input to the ML models.

V. Challenges with Data Management

In this section, insights into data management challenges are presented based on the findings from our cross-case analysis. We have identified ten major challenges with data management and existing data pipelines used in the company.

A. Data Availability

The availability of the right data in the right format at the right time is a basic requirement for the successful development of data products. Data collection is a difficult task and sometimes it fails due to authentication failure, environmental factors or failure of collection device. Even after collecting an enormous amount of data from the devices, it may not reach the intended destination. Data collected can be incomplete. i.e not all information will be available in the data warehouse. For instance, due to software failures, parts of the data can be lost. Unless we have a tracking mechanism, this loss is hard to identify. Furthermore, if well defined data is given as input to the model, it won't be able to give the same performance when unseen real-world data is given leading to underfitting.

B. Data Quality

For data-hungry systems like ML and DL, data quality is crucial. When low-quality data is fed to the algorithms, the low-quality output will be produced. For instance, while collecting fault logs from the devices in the field, there should be a clear distinction between faults due to environmental factors and faults due to device failure. The challenge is when the data is transformed to fit a predefined structure, "unnecessary" parts of it are removed. Therefore, we would need a method to save the original file. When the data is transformed on the fly and then stored, this would not be possible. It is always good to have the original raw data file stored so that it can be accessed whenever the structured data becomes insufficient to meet the requirements.

C. Data-flow Instability

Data-flow to the company's data storage is not always stable. The device at the company's end should be prepared to receive the data from the distributed devices. The pipeline, if existing, should be capable enough to handle data-flow through it. If multiple teams request data simultaneously, the pipeline should be able to serve it. Moreover, elements in the pipeline should be monitored properly for the failures. Failure of pipeline elements lead to data-flow instability. Timely upload of processed data is mandatory especially in case of dependencies. If data pipeline is accepting continuous, intermittent and batch data, at some points, the inflow of data will be heavy and during other times, it will have to handle only continuous data. This also lead to data-flow instability if the pipeline is not able to adjust its capacity.

D. Data Silos

A data silo is the gap between the data and the consumers who need the data. It is a result of poor architecture, legacy operational systems, and outdated company culture. The main problem is that the data becomes isolated and trapped without reaching the consumers. When individual teams develop their own data pipeline, it may also lead to data silos. There is a high probability that multiple teams performing the same activities for different use cases.

E. Data Dependencies

Data dependency occurs when a team or device has to depend on the outcome of some other team or device for starting their activity while developing any data product. There can have situations where the team has to wait for a long time for obtaining the required data. When the dependee is a software device that failed, the dependent won't be notified. Usually, in such cases after the expected time of delivery,

necessary actions are taken to check the existence of dependee. Data dependencies often lead to delayed production.

F. Data Pipeline Latency and Overhead

Data pipelines can create additional latency to the entire data workflow. Latency is defined as the time taken by data to travel through the entire pipeline. When multiple teams are requesting data or when the data inflow increases, the data pipeline may produce delayed outputs. Besides that, data has to go through all components and checks in the pipeline to reach the destination. Failure or slow down any one of these components in the transit can lead to latency. The data pipeline becomes an overhead when the data is used for cases for which data quality is not important.

G. Data Pipeline Owner Overloaded

The data pipeline owner is a person who is responsible for monitoring and managing the data flowing through the data pipeline. During peak times, that person gets overloaded. Moreover, it is always good to have maximum automation in the data pipeline.

H. Unreliable Data Pipelines

The reliability of a data pipeline depends on the reliability of its elements. Therefore, elements within a data pipeline should be made fault-tolerant. A data pipeline without built-in validation, mitigation mechanisms cannot ensure data quality.

I. Low Storage Capacity

When every team is constructing their own data pipeline, everyone stores the same data in different forms in the data storage leading to shortage of storage space. Databases, data warehouses, data lakes are for eliminating redundant storage of data. However, an increased number of data pipelines will eventually lead to reduced storage capacity. Another reason is the storage space division between teams. When storage is divided between teams, each one will get only a small portion of the actual available storage space.

VI. Data Pipeline Meta-model

The section above described challenges with the data management encountered by the industry practitioners. From the empirical findings and through analysis of existing pipelines, we develop a conceptual model of data pipeline that can potentially overcome the limitations of existing data pipelines.

Meta-model is a set of concepts used to build data pipelines. Nodes and connectors are the two main basic components used to build a data pipeline. Nodes are interconnected with each other using connectors. Both of the components have certain capabilities. For instance, the ability to connect different nodes is the capability of the connector. Fig. 5 shows the components, capabilities of both nodes and connectors, notations used to represent nodes and connectors, etc. Colour coding, icons, and differences in style are used to represent the variations in nodes and data that is flowing through the nodes. Capability is the ability of a node to perform a certain activity. For instance,

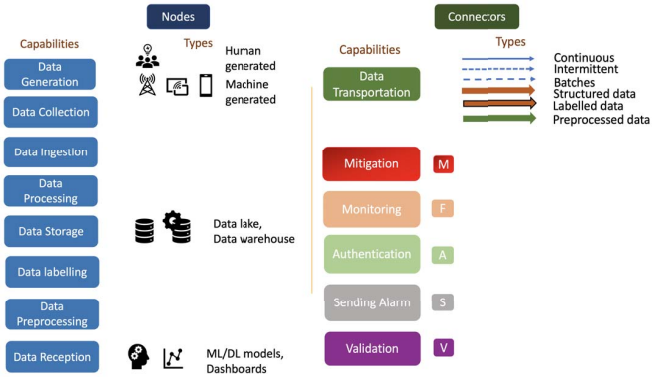

Fig. 5. Meta-model for building data pipeline

data sources in the pipeline have the capability to generate data.

Capabilities of Nodes: Data Generation, Data Collection, Data Ingestion, Data Storage, Data Processing, Data Labelling, Data Pre-processing, Data Reception.

Capabilities of Connectors: Data transmission, Authentication, Validation, Monitoring, Mitigation, Sending alarm.

The ability of the connectors to perform certain activity is termed as capability of connectors. Connectors have different capabilities. Connectors are the carriers of data. i.e they transmit data from one node to the next. Some connectors carry raw data and some carry processed data. Some connectors carry labeled data. Each of these connectors are given separate notation and color-coding in fig. 5. All connectors have the capability to transmit data. Apart from that, they have additional capabilities like authentication, validation, monitoring, mitigation and sending an alarm. The authentication capability of a connector is denoted by placing a light green color square with 'A' on top of it. Similarly, validation is represented by a yellow square with 'V' on it. Mitigation is represented by a red square with 'M'. Monitoring by the beige color square with 'F' on it. As the letter 'M' is already taken for mitigation, we use F(Fault detection). Grey color square with 'S' denotes the capability to send the alarm.

VII. Conceptual Model of Data Pipelines

Data pipeline is a complex series of components interconnected with each other where the output of one component is fed as input to the other. The starting point of the data pipeline is called source and the final destination is called a sink. All other nodes are intermediate nodes. Each component in the data pipeline manipulates data by performing activities. The data pipeline model presented in this section is a conceptual model. According to the requirement, it can be used by any organization for any data application by creating instances. Fig. 6 illustrates the conceptual model of a data pipeline.

Data Generation: Data pipelines start from a source and in most real life cases the source will be multiple and distributed. Therefore, our pipeline also has multiple sources distributed all around the world. Data sources can be of

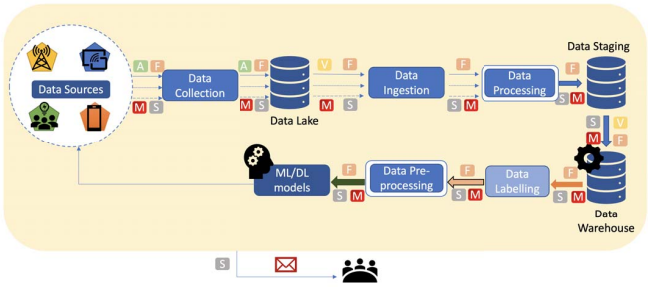

Fig. 6. Conceptual model of data pipeline

different types. Any device having the ability to generate data is called a data source. Data sources considered here are classified into two categories: Human-generated and Machine generated. Data that is produced through manipulations by team members, other team members or other organizations come under human-generated data. Machine-generated data are produced by the various devices employed in different equipment, vehicles and so on. For instance, data generated by a device embedded inside the car, data produced by mobile applications, etc.

Data Collection: The data-flow from the source can be batch, intermittent or continuous. Three different styles of arrows starting from data sources indicate that the connector between the data source and data ingestion can carry all the three variations of data-flow. Although the data collection node can collect data from the sources, it should show the permission to collect data from the sources.

Raw Data Storage - Data Lake: The data collected from the source will be raw and should be stored so that the original data files can be retrieved in the future. However, the data collection node has to show its right to ingest data into the data pipeline. This authentication is carried out by connectors between data collection and data lake.

Data Ingestion: Raw data files can be taken from the data lake and ingested into the data processing component. This data ingestion method will be different for different types of data. Data can arrive in all shapes and sizes. Real-time stream data will be processed sequentially. The continuous data will be validated immediately after extracting it from the data lake.

Data Processing: Data processing itself is a composite node in which there can be multiple individual components like data aggregation, data parsing, data transformation, etc. Data aggregation is the process by which raw data is expressed in a suitable form for statistical analysis. With data transformation, the unstructured aggregate data is converted into a structured format or semi-structured format. Thus, the data processing step converts all different types of data into a single format and stored in the data staging area. This is symbolically represented in the fig. 6 with three different incoming arrows to data processing indicating continuous, intermittent and batch data. The output from the data processing step is a single thick arrow.

Data Staging and Data Warehouse: The data staging is a temporary storage area where the data can be stored for validation. After validating the structured or semi-structured data, it is stored in a data warehouse. This data warehouse functions as a point from which the data can be taken for several data applications like report creation, ML applications, dashboard creation, etc.

Data Labeling: Our study is mainly focused on ML applications. Therefore, the data pipeline shows the necessary steps for automating the data pipeline for ML applications. ML algorithms can be supervised, unsupervised or reinforcement. For unsupervised algorithms, the data labeling step can be skipped. That is the reason why the data labeling step is shown in a light blue color different from the other nodes. As most of the companies are using a supervised approach for their ML applications, the focus is more on the same.

Data Pre-processing: To achieve better performance from ML algorithms, data needs to be pre-processed before training. The pre-processing depends on the practitioners developing the application and also the nature of the problem. Nevertheless, popular data pre-processing steps are data quality assessment, data imputation, data encoding, data sampling, dimensionality reduction, etc. Thus, data pre-processing can include any process which transforms data in such a way that it can be fed to a machine-learning algorithm.

Data Reception: The output from the data pre-processing is given as input to the ML models which uses it for training, retraining, testing, and validation. The ML model act as a data sink in fig. 6. As most of the companies are following agile methodology, the data will be collected back from the data products for further iterations. Therefore, data produced by these ML applications are again collected by the data collection node and goes through the pipeline continuously.

Capabilities of Connectors: Each connector between the nodes have the capability to send data to any other node, monitor the data-flow, fault detection, check for associated mitigation strategies when a fault is detected. If there is no defined mitigation strategy, then it has the capability to send alarms to the responsible team. The faults that may happen at each stage will be different. Therefore, mitigation strategies also will be different. Similarly, the responsible team/person who can handle a particular fault will be also different from each other.

To summarize, the conceptual model of the data pipeline is fully automated in which monitoring is performed throughout the pipeline. The data pipeline is fault-tolerant to some extent because mitigation strategies are there to ameliorate the effects of faults. Moreover, teams can request data from any point in the pipeline according to their requirement.

VIII. Validation Study

The conceptual data pipeline model was validated through interviews with 20 industry professionals from three companies of different domains and different maturity levels. Two of the authors, together with one author online conducted the interview study for validation. The purpose of the research was to develop a conceptual model of a fully automated,

fault-tolerant and traceable data pipeline. Table 2 illustrates the challenges with data management that can be partially or completely solved by the usage of data pipelines.

Data availability can be solved to a certain extent with the proposed data pipeline. However, when the source fails to generate data, then the data will not be available. The failure will be detected automatically and the corresponding mitigation strategies will be adopted. Also, when the customer is reluctant to share data, then, of course, the data will not reach the pipeline. In such a scenario, the corresponding manager will receive an alarm and they can take necessary actions to ameliorate the situation. Data quality challenges cannot be completely solved. If the data produced by the source is low quality, there is no inherent mechanism in the pipeline to make it high quality. However, high-quality data will not lose its quality during its transmission through the pipeline. Data-flow instability can be also solved using the proposed data pipeline with its built-in mechanisms to control the flow of data. Data silos is a complicated phenomenon that cannot be solved by the mere introduction of a data pipeline. It needs reorganizations, a cultural shift in the company, etc. Data dependencies can be completely solved as the pipeline itself is designed to be fully automated. Dependency between teams will be eliminated and all teams will have a dependency on the data pipeline. Data pipeline latency will be there. As the components in the pipeline are more and all those components have got intelligence in the form of capabilities, latency comes as a side effect. Data pipeline owner overhead can be reduced as the responsibility will be spread across multiple persons who have got a better understanding of a specific part of the data pipeline. The reliability of the data pipeline can be ensured with the connector level mitigation strategies. Failure of a particular component will affect the data-flow which will be detected by the monitoring mechanism and the fault is taken care of either by the mitigation strategies or by the corresponding responsible person. Storage capacity cannot be increased by implementing the data pipeline. As discussed earlier, it can eliminate the redundant storage of data. However, no provision in the data pipeline can increase storage capacity.

TABLE II
ANALYSIS OF DATA CHALLENGES THAT CAN BE SOLVED WITH THE PROPOSED DATA PIPELINE

Challenges with Data Management	Proposed Data Pipeline
Data Availability	Partially solve
Data Quality	Partially solve
Data-flow Instability	Completely solve
Data Silos	Cannot solve
Data Dependencies	Completely solve
Data Pipeline Latency	Cannot solve
Data Pipeline Owner Overloaded	Completely solve
Unreliable Data Pipeline	Completely solve
Low Storage Capacity	Cannot solve

The model developed by the first author was presented before the industrial experts and their consensus and disagreements were recorded. The validation section will be structured in terms of agreements and suggestions for improvement. Agreements refer to situations where practitioners agree and confirm while suggestions for improvement refer to situations in which the interviewees had a different opinion. Some minor corrections were made to the model and the other concerns raised are saved for extended work of the data pipeline model.

Case A: Manufacturing Company

The conceptual model was presented by the second author and the third author collected the feedback from the industry professional.

Agreements: The conceptual data pipeline model was identified as a standard concept that can be used by teams located in different parts of the world while building their data pipeline. With a standard architecture, there can have a common understanding of processes. Automation of monitoring and mitigation are interesting and important

Suggestions for Further Enhancements: Monitoring has three variations such as performance monitoring, data profile monitoring, and condition monitoring. Performance monitoring can continuously check for the software performances in the data pipeline. Condition monitoring can ensure the data pipeline health and data profile monitoring can make sure that the data flowing through the data pipeline meets the data quality requirements, detect faults and allows investigation of data problems thereby making it traceable.

Case B: Automobile Company

The conceptual model for data pipeline was presented by the first author and we collected the agreements and disagreements from all the industry professionals.

Agreements: Practitioners recognized this conceptual model as a language for communication between data professionals. When there is a common language, it is easy to avoid misinterpretations. Moreover, they all agreed about the monitoring spread across the pipeline and the need for different storage stages.

Suggestions for Further Enhancements: The major disagreement was concerning the nomenclature of nodes in the data pipeline. The data warehouse is a term to represent the aggregated and well-transformed data which is used for a specific use case. In the conceptual model, the data warehouse is capable of storing data for multiple applications. The suggestion was to rephrase all these ambiguous names such as data preparation and data processing. Another problem is that the model does not give attention to the reinforcement algorithm. Data pipeline does not explain who is the person responsible for each step of the pipeline. To increase readability and understandability for everyone in the company, it was suggested to have different model views. For instance, a model with no technical terms, a second one with much lesser abstraction and so on.

Case C: Telecommunication Company

Internal validation of the data pipeline is performed at the telecommunication company. The first author presented the work and recorded the responses from the team.

Agreements: The team liked the conceptual model of the data pipeline as they all were developing their pipeline for a particular use-case. They realized the need for a standard pipeline model that can be followed by everyone in the organization. Moreover, some of them were happy about storing original data in a data lake as they were experiencing issues with the availability of raw data files.

Suggestions for Further Enhancements: The data processing step itself can include data labeling and data preprocessing which is shown as separate steps in the proposed data pipeline model. They suggested it is good to have separate storage at each step of the data pipeline. There should be a provision to stop sending alarms continuously to the team whenever the issue persists for a longer duration.

IX. Threats to Validity

This study was based on existing data pipelines developed by different teams from the same company located in various parts of the world to reduce the bias of operating with a single team within the same organization. To address internal validity threat, one of the authors who has in-depth knowledge about the data process in the company, was asked to validate the findings. Also, the findings were validated with other teams in the company who were not involved in the study. Furthermore, the study was validated again by external companies from different domains.

X. Conclusion

In the immediate future, it will be inexorable that the daily analysis will not be able to keep up with the daily influx of data. Along with the increased popularity of data and its products, challenges associated with it also increased. Data scientists and other practitioners working with data spend a considerable amount of time combating with those challenges. The proposed data pipeline model can either solve or alleviate several data management challenges with limited human intervention. However, data pipelines need to be carefully designed so that data-flow can be monitored, managed and maintained. Therefore, fully automated, fault-tolerant and traceable data pipelines are gaining importance. The conceptual data pipeline model proposed in this paper has nodes and connectors which perform the activities in the data workflow. The conceptual model is validated using an exploratory case study where a total of 20 practitioners from three different companies participated. All of them agreed with the necessity of data pipelines in the organization, they liked the structuring of the conceptual model and the automation of monitoring, alarming and mitigation. They also gave a few suggestions for the improvement of the model. As future work, description on mitigation strategies at each step, the physical realization of the conceptual model will be done and the results will be analyzed.

Acknowledgments

This work is in part supported by Vinnova and by the Software Center. The authors would also like to express their gratitude for all the support provided by Ericsson.

References

[1] M. W. Van Alstyne, G. G. Parker, and S. P. Choudary, "Pipelines, platforms, and the new rules of strategy," *Harvard business review*, vol. 94, no. 4, pp. 54–62, 2016.

[2] R. Matheus, M. Janssen, and D. Maheshwari, "Data science empowering the public: Data-driven dashboards for transparent and accountable decision-making in smart cities," *Government Information Quarterly*, p. 101284, 2018.

[3] J. G. Stadler, K. Donlon, J. D. Siewert, T. Franken, and N. E. Lewis, "Improving the efficiency and ease of healthcare analysis through use of data visualization dashboards," *Big Data*, vol. 4, no. 2, pp. 129–135, 2016.

[4] G. Gautam and D. Yadav, "Sentiment analysis of twitter data using machine learning approaches and semantic analysis," in *2014 Seventh International Conference on Contemporary Computing (IC3)*. IEEE, 2014, pp. 437–442.

[5] S. M. S. Tanzil, W. Hoiles, and V. Krishnamurthy, "Adaptive scheme for caching youtube content in a cellular network: Machine learning approach," *IEEE Access*, vol. 5, pp. 5870–5881, 2017.

[6] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in *Proceedings of the 10th ACM conference on recommender systems*, 2016, pp. 191–198.

[7] L. Deng, J. Li, J.-T. Huang, K. Yao, D. Yu, F. Seide, M. Seltzer, G. Zweig, X. He, J. Williams *et al.*, "Recent advances in deep learning for speech research at microsoft," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 8604–8608.

[8] H. Sun, S. Hu, S. McIntosh, and Y. Cao, "Big data trip classification on the new york city taxi and uber sensor network," *Journal of Internet Technology*, vol. 19, no. 2, pp. 591–598, 2018.

[9] K. Goodhope, J. Koshy, J. Kreps, N. Narkhede, R. Park, J. Rao, and V. Y. Ye, "Building linkedin's real-time activity data pipeline." *IEEE Data Eng. Bull.*, vol. 35, no. 2, pp. 33–45, 2012.

[10] H. H. Olsson and J. Bosch, "From opinions to data-driven software r&d: a multi-case study on how to close the'open loop'problem," in *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications*. IEEE, 2014, pp. 9–16.

[11] M. Banko and E. Brill, "Scaling to very very large corpora for natural language disambiguation," in *Proceedings of the 39th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 2001, pp. 26–33.

[12] E. Deelman and A. Chervenak, "Data management challenges of data-intensive scientific workflows," in *2008 Eighth IEEE International Symposium on Cluster Computing and the Grid (CCGRID)*. IEEE, 2008, pp. 687–692.

[13] P. Vassiliadis, "A survey of extract–transform–load technology," *International Journal of Data Warehousing and Mining (IJDWM)*, vol. 5, no. 3, pp. 1–27, 2009.

[14] J. Trujillo and S. Luján-Mora, "A uml based approach for modeling etl processes in data warehouses," in *International Conference on Conceptual Modeling*. Springer, 2003, pp. 307–320.

[15] T. Rabl and H.-A. Jacobsen, "Big data generation," in *Specifying Big Data Benchmarks*. Springer, 2012, pp. 20–27.

[16] S. Amershi, A. Begel, C. Bird, R. DeLine, H. Gall, E. Kamar, N. Nagappan, B. Nushi, and T. Zimmermann, "Software engineering for machine learning: A case study," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 2019, pp. 291–300.

[17] J. M. Verner, J. Sampson, V. Tosic, N. A. Bakar, and B. A. Kitchenham, "Guidelines for industrially-based multiple case studies in software engineering," in *2009 Third International Conference on Research Challenges in Information Science*. IEEE, 2009, pp. 313–324.

[18] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical software engineering*, vol. 14, no. 2, p. 131, 2009.

[19] J. Singer, S. E. Sim, and T. C. Lethbridge, "Software engineering data collection for field studies," in *Guide to Advanced Empirical Software Engineering*. Springer, 2008, pp. 9–34.

[20] S. H. Khandkar, "Open coding," *University of Calgary*, vol. 23, p. 2009, 2009.