Integrated Analytics for IIoT Predictive Maintenance using IoT Big Data Cloud Systems

Hong-Linh Truong
Faculty of Informatics, TU Wien, Austria
E-mail: hong-linh.truong@tuwien.ac.at

***Abstract*—For predictive maintenance of equipment with Industrial Internet of Things (IIoT) technologies, existing IoT Cloud systems provide strong monitoring and data analysis capabilities for detecting and predicting status of equipment. However, we need to support complex interactions among different software components and human activities to provide an integrated analytics, as software algorithms alone cannot deal with the complexity and scale of data collection and analysis and the diversity of equipment, due to the difficulties of capturing and modeling uncertainties and domain knowledge in predictive maintenance. In this paper, we describe how we design and augment complex IoT big data cloud systems for integrated analytics of IIoT predictive maintenance. Our approach is to identify various complex interactions for solving system incidents together with relevant critical analytics results about equipment. We incorporate humans into various parts of complex IoT Cloud systems to enable situational data collection, services management, and data analytics. We leverage serverless functions, cloud services, and domain knowledge to support dynamic interactions between human and software for maintaining equipment. We use a real-world maintenance of Base Transceiver Stations to illustrate our engineering approach which we have prototyped with state-of-the art cloud and IoT technologies, such as Apache Nifi, Hadoop, Spark and Google Cloud Functions.**

I. Introduction

A sheer number of equipment to be managed in various industries requires full automatic systems to capture equipment operation status, analyze them, and automatically control these equipment. Therefore, in predictive maintenance [1], [2] we have observed a number of works that use state-of-the art Internet of Things (IoT) devices, scalable message brokers, and big data analytics in the cloud to monitor and analyze equipment, such as discussed in [3], [4]. However, it is also recognized by various researchers and practitioners that, in many cases, we cannot fully rely on software systems and intelligent algorithms to predict and maintain such equipment. Several approaches have been discussed, such as using crowd-sourcing tasks or improving artificial intelligence with humans [5], [6], in order to deal with complex interactions by combining software and human capabilities.

We focus on Industrial Internet of Things (IIoT) with predictive maintenance functions built atop *big data analysis and expert capabilities*. Considering common maintenance workflows with big data analytics pipelines and services in IoT Cloud systems, we identify important places where insights about system problems and equipment problems will be communicated to humans and services, utilizing human capabilities to decide and address possible activities in predictive maintenance. While other researches focus on what humans can do in terms of data analytics and data gathering [7], [8], [9], we focus on how we enable humans to be in the loop through service design and engineering perspective, but we do not concentrate on managing and optimizing what humans will do. We believe that this is a fundamental issue that has been under-researched in complex IIoT.

To enable complex interactions among software and human in predictive maintenance, we leverage reactive software design principles with serverless technologies to allow extensible and fast integration between functions verifying maintenance indicators and tasks to be carried out for such indicators. Furthermore, we address the interdependencies among ICT system incidents (potential problems of the IIoT-based big data analytics) and equipment incidents (potential problems of equipment to be maintained) in an integrated manner, as we believe the accuracy and effectiveness of predictive maintenance at a large scale cannot just deal with these two sectors in a different way. For example, human capabilities are used not only to verify status of equipment but also to control sensors and suitable data analytics for deciding which maintenance tasks should be done. Overall, this paper contributes to service engineering techniques for operations that humans play important roles:

- novel architecture and detailed analysis of system incidents and critical analytics of equipment in IIoT integrated analytics,
- holistic task analytics generation and integration for both software and humans in IIoT maintenance using function as a service, and
- human interactions in IIoT for (i) performing maintenance of equipment based on *critical analytics results* and (ii) controls the IoT Cloud systems for equipment analytics based on *system incidents*.

For a proof-of-concept, we will describe various examples from software design perspectives with the real software prototype based on the maintenance of infrastructures of a network of Base Transceiver Stations (BTSs).

The rest of this paper is organized as follows: Section II present an analysis of complex interactions in IIoT predictive maintenance. Section III details how we augment IoT Cloud systems with holistic tasks for software for predictive maintenance. Section IV details human interactions and tasks. We

present our prototype and illustrating examples in Section V. Related work is discussed in Section VI. We conclude the paper and outline our future work in Section VII.

II. Analysis of Complex Interactions in IIoT Predictive Maintenance

A. IoT Cloud systems for IIoT Predictive maintenance

To maintain a large number of equipment from various manufacturers, commonly we rely on different sensors, actuators, gateways, cloud storage, data processing algorithms and intensive domain-knowledge. In particular, the sensor and actuator parts are specifically designed to interface to equipment to monitor and control the equipment. However, other software and system parts are quite common to enable large-scale data collection and analytics. Typically, the main goals of IIoT predictive maintenance are (i) to use streaming analysis to detect early warning situations and (ii) to gather historical data for batch analytics, which can provide more insights into equipment problems. State of the art tools are usually built atop various big data services and scalable message brokers, such as MQTT Mosquitto, IoT Hub (e.g., from Google, Microsoft Azure, and Amazon EC2), BigQuery, Apache Hadoop, Apache Spark, Apache Kafka, Apache Flink, to name just a few. However, for predictive maintenance, domain knowledge is also crucial. Therefore, each third party in predictive maintenance usually has its own way of handling situations based on results from data analytics and its domain expertise.

Figure 1 presents an IIoT system that we have built for maintaining equipment in Base Transceiver Stations (BTSs) and tested in research pilots with several BTSs in Vietnam. In a BTS, we have sensors, actuators and gateways to monitor and control equipment, such as power generator, backup battery, HVAC, grid electricity sources, etc. BTS infrastructure maintenance companies have private clouds to gather equipment monitoring data: these companies can share the data to the telco company (BTS owners) as well as other third-party companies doing the maintenance for specific equipment. In general, maintenance companies and the telco company have infrastructures for *Data Sources* to gather monitoring data, covering status of equipment and performance of telco network nodes (e.g., NodeB). Such data sources are shared to various services in *Predictive Maintenance Analytics*, which are built atop Apache Nifi, Hadoop FS and Spark (using Google Dataproc), Google Big Query, Elastic Search and Kibana, Apache Kafka and Apache Flink.

The above-mentioned system has several common components and interactions that we see in typical IoT Cloud systems. First, IoT subsystems are used for monitoring and controlling equipment. Second, enterprise Cloud subsystems for data processing, analytics and storage. Except the infrastructure services, these services can be on private or public cloud but they are completely owned by the telcos and predictive maintenance companies due to the sensitivity of data to be processed. With current virtualization and service engineering techniques, some important aspects we have observed are:

- Sensors and actuators: to support different data gathering capabilities, sensors can be reconfigured at runtime.
- Data Sources: several data sources are available, Besides sensoring data sent through gateways, we also have data sources from other parts of the BTSs (e.g., telco NodeB) and from extracted sensoring data. However, sharing such data sources can be based on requests from analytics triggered by humans.
- Cloud data processing and analytics: various analytic pipelines are available, especially for big historical data. To save cost they can be deployed and undeployed on-demand. They include various cloud services to deal with data forwarding, storage and processing. For data analytics, typically we have streaming analytics and periodically pre-configured cron jobs of batch analytics; they are used for short term view on maintenance indicators of equipment. Many batch processing algorithms are executed only on-demand for in-depth equipment analytics.

Since the software systems are complex, IoT Cloud systems also are supported by service management systems, which perform typical IT tasks, such as controlling sensors and data analytics. Human interactions are thus required for IoT Cloud system maintenance, besides human interactions for investigating industrial equipment being maintained.

B. Integrated System and Application Domain Maintenance

1) IIoT and systems incidents: For predictive maintenance, we rely on various near real time and historical data sources. Many of them are collected by sensors from various places and transferred to the cloud through complex software systems. First, it is well-known that IoT data can have many data quality problems [10], [11]. Second, since the system is complex, various system incidents may occur across layers and subsystems [12], such as failed algorithms due to a bad data quality.

For all of these incidents, related to data, algorithms and system services in the IoT Cloud systems, we call them *system incidents* since they are not related to the equipment being maintained but the IoT big data systems supporting the maintenance. System incidents are in the focus on the IoT Cloud operation management and are usually treated differently from problems associated with industrial equipment and by different teams. However, in our view, such incidents should be also treated together with possible problems detected in equipment (analyzed from the maintenance application domain view). The rationale is that, for example, any issue in IoT data could lead to wrong analytics of equipment as well as could prevent further analytics of equipment. All of these are part of integrated predictive maintenance tasks.

2) Equipment status and critical analytics result: The key objective of predictive maintenance is to identify indicators of equipment, e.g., discussed in [13], which are used to trigger the maintenance of equipment. While conceptually we can list different types of indicators, such indicators are determined only through specific algorithms and data analytics together with domain experts. In our system, we have two

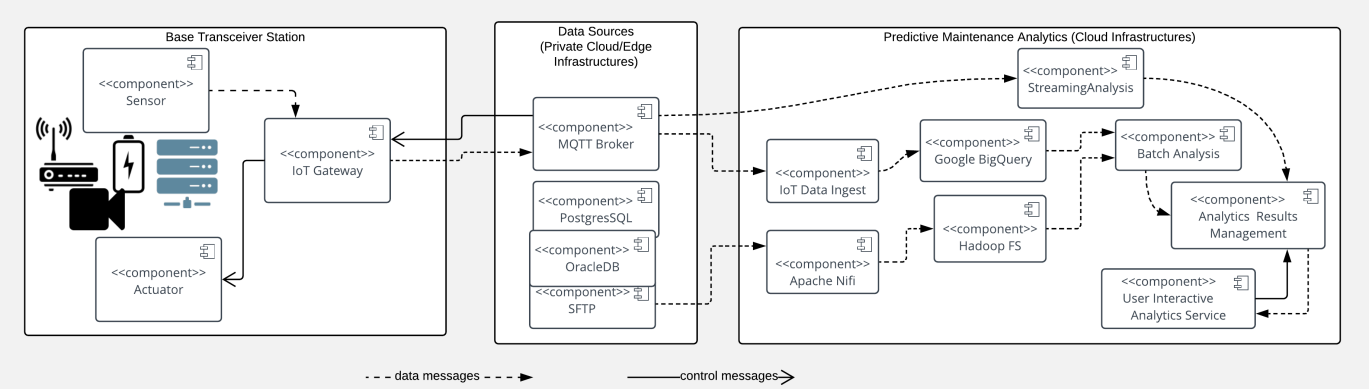

Fig. 1. BTS, data sources and services for predictive maintenance

steps: algorithms and data analytics present possible indicators encapsulated in what we call *critical analytics results*, then these results will be propagated into the right components, whether software or humans for further consideration. Thus, one critical analytic result can lead to the trigger of another further analytics. Our goal in this paper is not focused on specific indicators (and how to determine them) but the *interactions* among components in handling critical analytics results. One important issue is that we consider the system incidents and critical analytics results together as system incidents may strongly influence the analytics producing the result leading to equipment incidents.

C. Human interactions

Naturally, the ultimate goal is to reduce as many as possible human interactions by using (machine learning) algorithms. For example, one algorithm could, theoretically, predict a power generator status by analyzing its alarms and decide when a maintenance should be done. In practice, they are extremely challenging and still we need to support human interactions. This requires us to (re)design complex IoT Cloud systems to accommodate humans interactions. We also have to consider that, for complex maintenance, human capabilities are not pre-defined (static assignment); they have to be provisioned on-demand.

One of the first human interactions is to control and validate IoT data collection processes. For example, more data sources might be needed for analytics, but the data obtained might have quality problems. Second, monitoring data from sensors will be analyzed by different data analytics functions; each function can be described in a form of data analytics pipelines. Which functions and their parameters to be deployed are heavily relied on domain knowledge and we can have different functions for different purposes. Therefore, another type of interactions is to allow humans to change and deploy these functions on the fly, based on certain types of *critical analytics results* of equipment operation and *system incidents* of IoT Cloud systems. Concretely, we must distinguish different tasks that humans can solve within predictive maintenance:

- *change/configure sensors for data collections*: humans observe analytics results (e.g., data quality measured by analytics processes) and decide that more sensoring data are necessary for improving analytics. For this, humans can select specific types of new sensors to be deployed (e.g., with specific configurations), and modify configurations of existing sensors.
- *controlling cloud resources for handling/sharing collected data*: in certain situation, humans know better to prepare the cloud resources for handling collected data. In this case, humans can control cloud resources in advance, instead of relying on reactive control rules of clouds.
- *selecting and deploy suitable data analytics*: humans receive certain types of critical analytics results of equipment that the analytics system does not know which types of analytics should be done next to examine the problem. In this case, humans can select and deploy or reconfigure suitable analytic functions based on their expertise.
- *optimizing equipment*: humans might decide which parameters are the best in order to control equipment to fix equipment through its actuators.
- *fixing the physical systems of equipment*: suitable people can fix equipment physically.

The next important human interaction is actually to allow a transparent communication of tasks with humans w.r.t. requests and results. Both *Streaming Analysis* and *Batch Analysis* put analytics results into *Analytics Results Management* database and storage. Besides storing results of analytics in database and dashboards, for people decide what to do next, we need to communicate potential issues, derived from analytics results, to human computation systems which automatically determine right experts and enable automatic human tasks assignment. In this view, it is important to define critical analytics results and system incidents that trigger human tasks (besides invocations of software services).

III. Holistic Task Analytics and Generation

A. System Overview

Figure 2 elaborates and augments the architecture in Figure 1 with integrated integration of humans and software in IIoT predictive maintenance systems. First, we identify points of instrumentation where *critical analytics results* indicating issues of equipment and *system incidents* impacting the data analytics to be communicated to software services and to humans via human tasks. Such results and incidents will be delivered

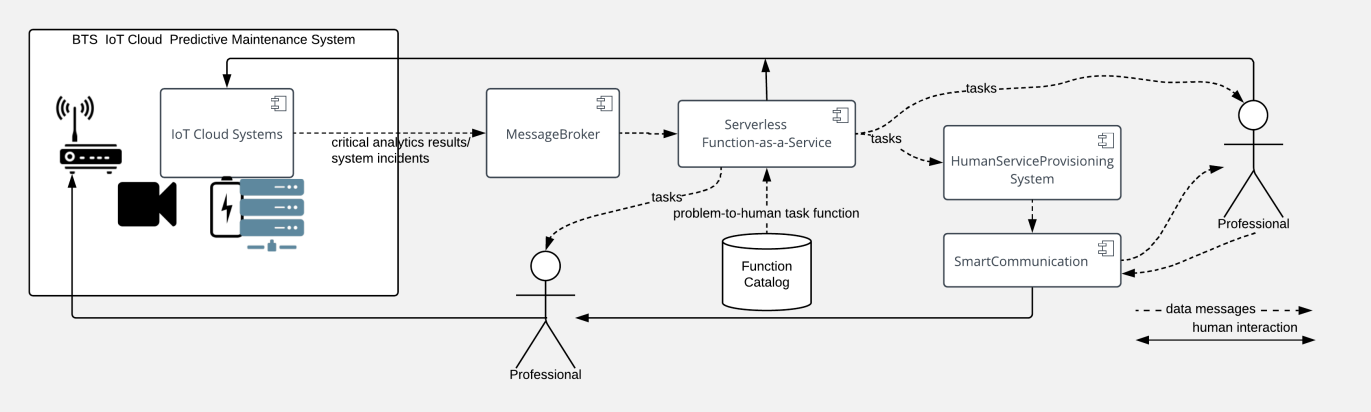

Fig. 2. IoT & Analytics for Predictive Maintenance

through a scalable message broker. A set of functions, in *Function Catalog*, will be triggered based on such results and incidents. Such functions will create human tasks that will be sent to *HumanServiceProvisioning Systems* (HSPS). HSPS will determine suitable professionals to perform the tasks and communicate the tasks to the professional via *SmartCommunication*. Professionals take the tasks and then perform suitable actions, such as control sensors (e.g., by using apps that call Webhooks/REST of services), deploy new data analytics functions, etc.

The key issue of the above-mentioned system is that several domain knowledge about predictive maintenance algorithms, equipment and professionals are specific. Thus, we need customized solutions for integrating them. We focus on service engineering aspects and the use of severless functions to create tasks that can be easily extended. To enable the integration between different professionals, we use *MessageBroker* to enable communication with different instances of HSPS, each instance may be provisioned for a maintenance company. *SmartCommunication* also enables different communication means for humans. In our current prototype, we use RabbitMQ as the MessageBroker, Google Functions for function-as-a-service, and RAHYMS[1] for HPSP. The communication is based on utilities using conversational cloud tools (e.g., SendBird and Applozic).

B. Points of instrumentation

Points of instrumentation (PoI) are where in an IoT Cloud system we capture information for generating tasks for possible maintenance activities. Technically, we can instrument the code (e.g., using a library to evaluate the analytics results at the end of the `PySpark` program), intercept the data flow (e.g., using a Nifi Processor implemented as an `ExecuteScript` to monitor quality of input data), or event-trigger evaluation (e.g., a function checking the analytics results when they are uploaded into Google Storage). We support these types of PoI.

In the architecture shown in Figure 2 we have not presented PoI for obtaining critical analytics results and system incidents. To identify such PoI, it is important to note that they are dependent on specific predictive maintenance systems. Given IoT Cloud systems for predictive maintenance shown in Figure 1, the following PoI have the usual flows and the point to report to other software services and humans:

- critical analytics results: warning, critical status, etc., detected from analytics results of equipment.
- system incidents: problems of quality of data in data collection, services failure, etc.

Within PoI, we need to write the logic to determine if a result or an incident requires software services or human tasks. We focus more on the interface and the message structure for PoI and leave the logic to be user-defined functions.

C. Reactive operations and extensible serverless functions

For reporting system incidents and critical analytics results to trigger other tasks, we leverage messages and reactive and serverless. Shown in Figure 2, instrumented code will create events, which triggers serverless function that generate suitable tasks for the next steps. The reactive principles[2] are important for this scale. For this maintenance, it is more than just waiting for report from algorithms periodically and let the human decides. We need to identify various situations:

- from the application domain based on critical analytics results: the outcome from one analytics could trigger other analytics automatically.
- from system operation based on system incidents: the incident of a system would trigger humans to look at potential issues in predictive maintenance analytics.

Combined with many tasks required human involvement, we need a flexible way to easily add, remove, and change functions which are loosely decoupled with other services and professionals for maintenance. We choose serverless [14] model as a way to integration to allow us to focus on events and functions that can be supplied by various stakeholders. Furthermore, choosing serverless function also enables flexibility and extensibility in developing functions performing mapping and invocation of issues to software or human tasks. To date serverless has been widely used in various domains [15] but its use in predictive maintenance with big data has not been

[1]https://github.com/tuwiendsg/RAHYMS

[2]https://www.reactivemanifesto.org/

seen. With serverless functions, various human capabilities in principle are function that we need to invoke through the right interactions. Serverless functions can be used to implement to trigger both interfaces for humans and integrated with IoT, big data and cloud systems directly. Second, functions can also be used to develop mappings of events to tasks and such mappings are changed other the time.

In principle, we can focus on writing functions that glue output and input of various services and algorithms in IIoT big data cloud systems, shown in Figure 1. Currently, we consider this work done by through software integration: function calls can be inserted during the software development and operation. Such functions are invoked based on events of system incidents and critical analytics results.

IV. Integration of Big Data Analytics with Human Tasks

A. Mapping incident records to human tasks

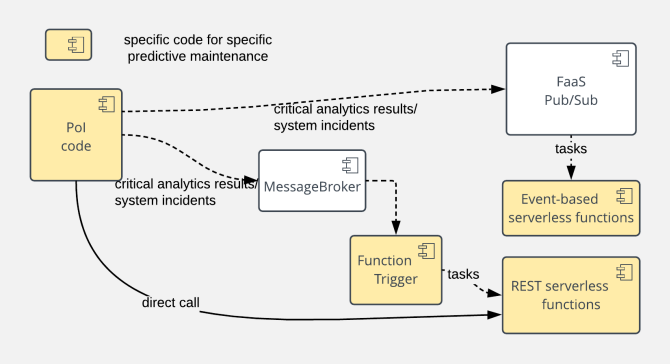

Fig. 3. Detailed serverless functions

From Figure 2, we have functions to be continuously developed and deployed into the *Function Catalog*. Sources of triggers are system incidents and critical analytics results from streaming analytics or batch analytics. The output of a function performed by humans is fed into, e.g., cloud deployment of services, control commands of devices, and messages to other humans. Events will trigger invocations of suitable functions producing human tasks, which will then be submitted to HSPS that can relay the tasks to humans. Figure 3 shows components in our focus for predictive maintenance for specific domains by leveraging generic serverless platforms in the cloud.

Events are triggers that are from various sources. Events related to IoT Cloud systems come from cloud monitoring, such as errors detected in Apache Nifi and failure of analytics algorithms from Apache Spark. Events related to BTS equipment are based on output of analytics, e.g., the streaming analytics processing detect some warning indicators about equipment status. Functions are continue to be developed and added into systems. This allows us to support extensibility of maintenance functions.

B. Human Task Function Catalog

As mentioned, human tasks are needed for critical analytics results of equipment as well as for system incidents. Figure 4 shows how human capabilities are integrated with IoT Cloud systems for predictive maintenance through catalogs of functions mapping human expertise and system incidents and critical analytics results. As the top level we have IoT systems and BTS equipment. At the second layer we have tasks (i) for controlling the IoT systems – there tasks aim to tune the IoT Cloud systems for dealing with data and analytics – and (ii) for equipment maintenance, dealing with issues related to equipment, based on analytics results. Such functions are based on specific domains, however, their structure and model are generic (our focus is on the structure and model). For the Function Catalog, currently, we use MongoDB to manage documents about the catalog. Besides other information, these documents record the relationships between (`incident`, `function name`) whereas `incident` indicates types of incidents and `function name` is used to identify the function which will be invoked to create human tasks.

Generally, such a task catalog will be extended to accommodate new development and changes of PoI and functions for mapping problems to human tasks. In our prototype, a PoI code will send to functions key information shown in Listing 1. The structure of `IncidentRecord` can be easily extended and customized to suit with predictive maintenance requirements as well as with other human task programming languages.

Listing 1. Task structure description

```
"IncidentRecord": {
    "name": "indicate the name",
    "id": "indicate the id",
    "log": "human readable content of tasks",
    "incident": "indicate the type of incident
        ",
    "severity":"indicate the severity of the
        incident",
    "tags":"possible tags about the incident",
    "meta": {
      "attribute_name":"value"
    }
  }
```

When a function receives an `IncidentRecord`, it will perform the mapping of the record to suitable task models of the underlying HPSP.

C. Scheduling Tasks to Humans

One important issue is to integrate with HSPS for solving problems with human tasks carried out by a collective of professionals (here we call a collective); in certain cases the collective might include only 1 professional at the beginning and its can be elastic during runtime. We note that this maintenance problem is not carried out by crowds but professionals from third party maintenance companies. When an analytics or a system monitoring concludes that a critical situation occurs and requires human intervention to investigate or to fix an incident, a human task request is created. This task contains a detail description of the situation, as well as a tag, which defines the kind of situations and a severity of the task. Shown in Figure 5, our work is to utilize existing human services

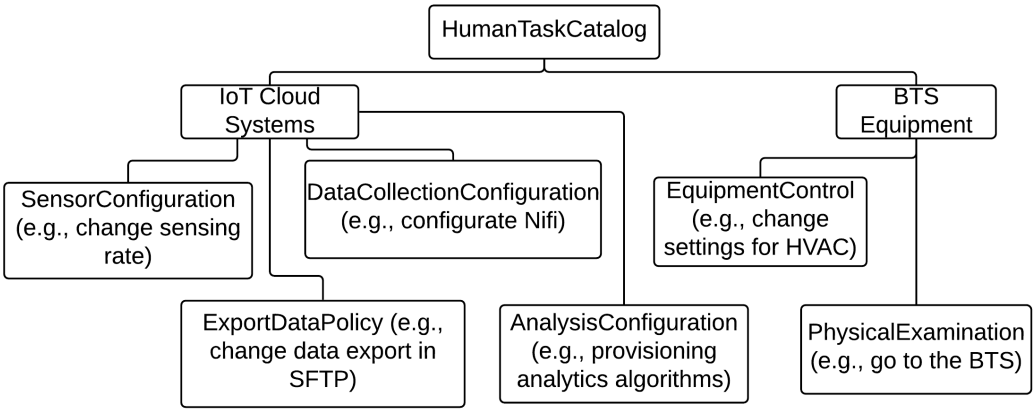

Fig. 4. Professional tasks and the function catalog

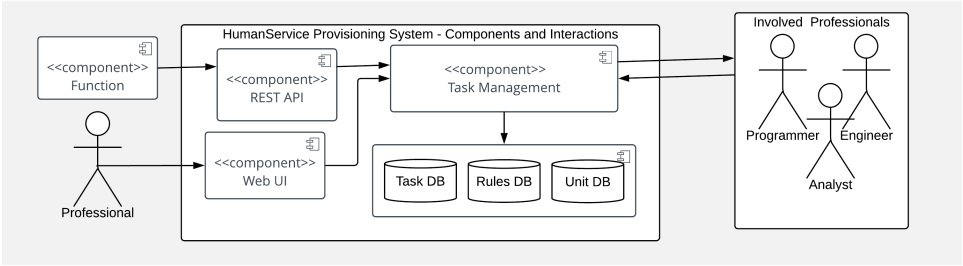

Fig. 5. Interactions within HumanServiceProvisioning Systems (HSPS)

which perform task assignments[3]. In terms of concepts and implementation, existing HSPS are quite complex.

1) Integration with RAHYMS: In this work, we explain key issues for integrating with HSPS through examples of our integration with RAHYMS, which is our HSPS in the prototype. RAHYMS provides runtime components and analytic tools for establishing a collective of professionals or individuals [19]. RAHYMS exposes its functionalities through client interfaces that can be used to manage tasks, to manage units (humans) performing the tasks, and to define a set of domain-specific rules that controls the composition of professionals for executing the task. Internally, when a task request is received, RAHYMS finds an appropriate rule on the DB matching the tasks tag and severity. The rule maps the task to a set of required humans to execute the task. Based on that RAHYMS composes professionals that provides the required services for the task.

Figure 6 presents basic descriptions for task assignment and service units within RAHYMS (with our minor extension). For solving a complex task, `Task` submitted by functions, RAHYMS creates a collective `Collective`. A collective has many professionals, each described by `Unit` and the number of professionals can be dynamic changed. `Unit` describes basic information about a professional that can offer certain services for Predictive Maintenance. It also includes information for interacting with the professional. Note that additional properties of Units can also be specified on the consequence part (e.g., skill level, load factor, etc) for enabling collective compositions. The key information for making the decision of the Collective and Professional is the `TaskRule`, which is domain specific knowledge about the type of incident indicators and severity and the expertise. All of these descriptions are HSPS specific whereas domain knowledge will help to create the right instance of such descriptions. Since we use RAHYMS these descriptions are built atop RAHYMS.

2) Principles for integration with existing HSPS: We can have a generic way for integrating with other systems that provide similar mechanisms to interact with professionals. Two important points for us to support complex interactions with humans, who are often managed and provisioned by external systems of third party companies, are

- for integrating with other services in IoT Cloud systems, HSPS must provide well-defined APIs, e.g., REST API calls and task structure (see Listing 1),
- HSPS must allow domain-specific knowledge to be defined, e.g., rules and human specifications with BTS predictive maintenance and IoT Cloud systems in our case. Without such features, it is difficult to utilize existing HSPS for predictive maintenance.

Note that in our work, we assume the real human will perform the task. In principle agent acting on behalf of humans (e.g., https://github.com/huginn/huginn) can also be investigated.

[3]Task assignment is a complex problem with many papers, such as [16], [17], [18]

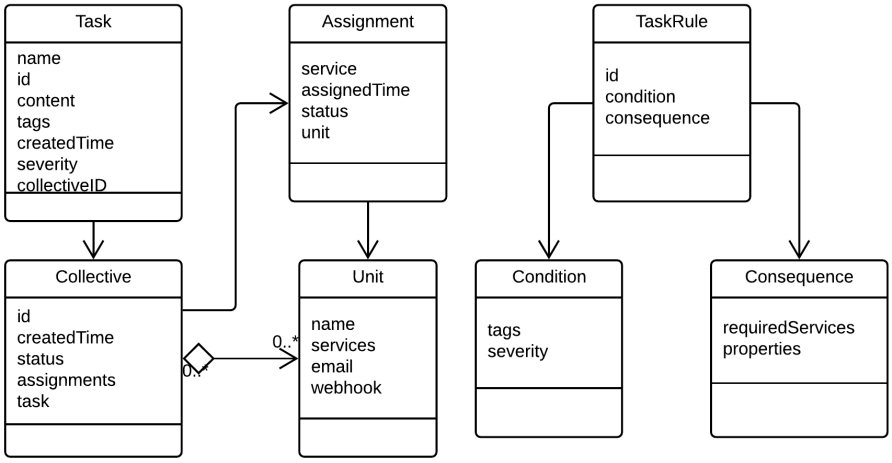

Fig. 6. Data models for assigning tasks to professionals based on RAHYMS

D. Communications with Humans

A *SmartCommunication* (see Figure 2) is used to exchange messages among humans and from software services to humans in IoT Cloud systems. We use common techniques in conversation, such as emails, chats, and video through webhooks, emails and mobile apps. We can also use other cloud services, such as Slack to transfer messages among professionals. Furthermore, it is also possible to use (i) *SmartCommunication* as a channel for reporting tasks that the HSPS can receive tasks and schedule the collective, or (ii) integrated industry systems for incident management systems, such as OpsGenie [20], for interacting with humans. However, this feature is not in the focus of this paper.

E. Instantiating multiple HSPS

As we see in complex interactions in predictive maintenance, in principle, we need to interface to multiple HSPS because the predictive maintenance for different equipment might come from different companies. Such instances can rely on a single HSPS infrastructure, which supports a multi-tenant model, or on different HSPS infrastructures.

V. Prototypes and Examples

We have implemented a proof-of-concept of our prototypes by utilizing Apache Spark, HDFS, AMQP, Google PubSub, serverless framework with Google Cloud Functions, and RAHYMS. In this section we present some examples to illustrate the engineering of our integrated analytics[4].

A. Capturing system incidents related to data movement

One example is to move data from the third-party maintenance company to the cloud for analytics of equipment. Using Apache Nifi, the data movement, e.g., from SFTP sources to Hadoop FS, is seamlessly integrated. However, a system incident might happy to the data file, e.g., data is not complete due to missing files. A quick check within Nifi will trigger an event that informs a person from the maintenance company to examine the issue. The following code illustrates an example of *a PoI in Nifi sending a system incident* through RabbitMQ.

```
var flowFile = session.get();
var company ="REMOVED";
var sender  ="DCIS-NIFI";
if (flowFile !=null) {
  //check if we have the expected data
  //...
  // we also customize  attributes for a
     specific company

    var incidentrecord = {
        "incident": "missingdata",
        "severity": "WARNING",
        "meta": {
          "absolutehdfspath":absolutehdfspath,
          "company": company,
          "sender" : sender,
          "occurredAt" : occurredAt
                        }
    };

    //send event to the output stream which is
         connected to RabbitMQ output
    flowFile = session.write(flowFile, new
        OutputStreamCallback(function(
        outputStream) {
        outputStream.write(JSON.stringify(
            incidentrecord, null, "  ").
            getBytes(StandardCharsets.UTF_8));
    }));
   //other code
}
```

B. Triggering in-depth analytics based on streaming analytics

One example is to trigger in-depth batch analytics of several months of historical data based on *critical analytics results* produced by a streaming analytics. Based on a window of time of events about alarms of a power generator, a critical analytics result is generated and sent to the message queue.

[4]We use the real implementation to illustrate examples. However, due to sensitive reasons in industrial collaboration, we adapted code for examples to reflect ideas rather than to show the real code.

A trigger component listening the queue decides to generate a new event for triggering a function that call an analytics web service to analyze power generators of stations over the last 6 months:

```
  //...
  //based on input event
  // determine analytics name and problem, e.g
      .,
  analytics_name ="powergenerator-alarm";
  analytics_problem="
      powergenerator_alarm_frequency";
  //prepare requests
  analyticRequest = {
         "companyId":companyId,
         "stationId":list_of_stations,
          "startTime":last6month,
          "endTime": occurredAt,
         "dataSource":"hdfs://.../",
         "outputFormat":"csv",
  }
  //call remote service
  rest_service_call =analyticsservice_url+"/"+
      analytics_name+"/"+analytics_problem;
  var options = {
                method: 'post',
                body: analyticRequest,
                json: true,
                url: rest_service_call
  }
  request(options, function (err, res, body) {
    //.. call remote web services
  });
  //...
 }
```

The in-depth analytics will produce the results that can be notified to the human (e.g., through company id, we store the result in Google Storage and inform the human via emails). Usually such in-depth analytics is carried out only on-demand, due to the cost of running it. This example shows how we can link different types of predictive maintenance analytics.

C. Functions for problem-to-human task

One example is to intercept the instance of Apache Nifi that collects data from *Data Sources* to trigger a human task to check the data when the quality is bad. The following excerpt shows the function triggered when data quality is bad:

```
//perform mapping ....
var tags =determine_tags(...);
//.... call relevant functions to determine
    name and content
//create human task
var humantask ={
      "name":task_name,
      "content":task_content,
      "tag":tags,
      "severity": severity,
       "meta": {
 //...
             }
};
//send humantask to HumanServiceProvisioning
//...
```

Another example for *critical analytics results* is to instrument PySpark analytics and create a human task when there are many alarms (more than usual) determined in a batch analytics. For example, the following excerpt shows an event sent from PySpark which runs the analytics for hourly data in HDFS:

```
"analyticsname": "AirConditionerAlarm",
"problem":"EXCEEDINGALARMS",
"meta": {
 "source":"hdfs://REMOVED/csvdata/
     station_alarm_data",
 "date": "2017-10-19",
 "hour": "12",
 "outputpath": "gs://REMOVED/
     test1_analytics2_20171020"
}
```

Based on that, the following excerpt shows a function that creates a human task, including various type of metadata about the data source and results (in an advanced setting, a similar function can be used to tell a human to deploy a new analytics if needed):

```
//determining content for human tasks
//....
//create human task
var humantask ={
  "name":"Run␣CLUSTERING␣for␣HVAC",
  "content":"Many␣alarms␣occurred␣for␣many␣
      stations.␣DataSource=hdfs://REMOVED.␣
      AnalyticsResult=gs://REMOVED",
  "tag":tags,
  "severity": severity,
  "meta": meta
};
console.log(humantask)
//submit human tasks
```

D. Examples of Professionals and Tasks

Listing 2 shows examples of rules that one needs to define for specific predictive maintenance deployment. They are used by HSPS to determine professionals and schedule tasks. Listing 3 and Figure 7 present examples of the output of HSPS algorithms to determine suitable professionals and put them into suitable collectives for performing tasks[5]. Figure 7 presents a sample of GUI through which professionals can accept, reject or delegate the task assignment. Given metadata, humans can also follow other resources, such as examining data in Hadoop FS and analytics results in Google Storage. As mentioned, our work relies on RAHYMS so the scheduling accuracy and communication for human tasks are based on RAHYMS.

Listing 2. Rules for handling missing data and HVAC

```
{
  "conditions": {
  "tag": "logdata_missing",
  "severity": "CRITICAL"
  },
```

[5]The listing and figure present emulated results in which we removed all real personal and real deployment information in our tests

```
  "consequences": {
    "services_required": ["data_analytics","
        bts_operator","cloud_operator"],
  }
},
{
  "conditions": {
     "tag": "hvac",
     "severity": "CRITICAL"
   },
   "consequences": {
      "services_required": ["data_analytics",
          "hvac_analytics", "bts_maintenance",
           "bts_operator"],
   }
}
```

Listing 3. Messages sent by HSPS to professionals

```
Sending message to kt@iotcloud.removed.com:
You got a task:<br>
Task #1<br>
Tag: logdata_missing<br>
Severity: CRITICAL<br>
Data collection problems. DataSource=sftp://
    REMOVED. ApacheNifi=https://REMOVED/nifi.<
    br><br>
<a href='http://REMOVE:8080/web-ui/assignment.
    html#/status/1/1'>See details</a>
...
Sending message to dp@pm.removed.com:
You got a task:<br>
Task #2<br>
Tag: hvac<br>
Severity: CRITICAL<br>
Many alarms occurred for many stations.
    DataSource=hdfs://REMOVED. AnalyticsResult
    =gs://REMOVED<br><br>
...
Sending message to cv@telco.removed.com:
You got a task:<br>
Task #2<br>
Tag: hvac<br>
Severity: CRITICAL<br>
```

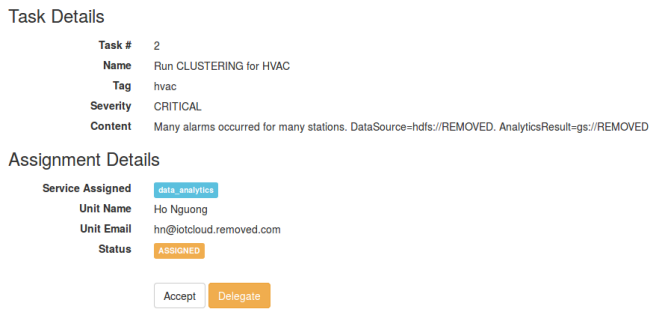

Fig. 7. Task assignments received by professionals visualized in RAHYMS

VI. Related Work

For architecture and software design, the work in [13] presents an "open architecture" for predictive maintenance but the architecture is not based on big data and IIoT. Also it does not incorporate novel features like serverless functions and human tasks. The paper [1] shows how to use mobile agents for for predictive maintenance. It presents the whole system which differs from ours w.r.t application domain, architecture and human interactions. Other systems like [21] include many components for predictive maintenance but not on the big data systems at large-scale like ours. The work in [22] shows the use of big data analytics for agricultural processes but they do not address maintenance and human tasks.

Many papers discuss about interactions in maintenance that require human and software [23], [24]. However, our work differs from these papers in two aspects: (i) they do not focus on big data systems pipelines cross layers for different kind of incidents, and (ii)they are not really leverage the human-as-a-service (although our work is just focusing on integration), we need to carefully design tasks. Many algorithms and case studies have been developed for predictive maintenance [2], [25], [26]. We also develop some but they are not in the focus of our paper. In general, we can leverage existing algorithms as long as they fit with equipment and our domain.

Many crowdsourcing papers have presented human tasks to collect data and evaluate data as well as perform data processing tasks [7]. Our paper is focused on critical business when only professionals are involved for predictive maintenance. In this view, our work addresses "hard tasks" required humans and machines [27]. But we are not aware of similar work when professionals are involved in predictive maintenance using big data analytics with IoT Cloud systems. Many authors present human-machine integration for data integration, data validation, and workflow [28]. Although it is related, our work is focused on professionals for analyzing and fixing problems due to issues of equipment operations detected by big data analytics. Business process and predictive maintenance utilize a lot of big data. But these systems do not discuss human-in-the-loop. In business workflow, the invocation of human tasks are popular but typically workflows are designed in advance.

A key difference of our work is about software design to enable various customization of human-in-the-loop. For example, the use of serverless functions for example enables us to create many way to generate human tasks in a DevOps fashion. Various human task programming languages, such as [29], [30], are related as they can be used by our systems for executing human tasks, although to our best knowledge they have not been used for professional work in IIoT.

VII. Conclusions and Future Work

Complex interactions between software and humans for IIoT predictive maintenance need to be captured and implemented in an extensible way. The goal of this paper is to analyze general interactions for integrated analytics and to present software design and engineering aspects. In this paper, we have addressed both system incidents and critical analytics results about equipment in an integrated manner for IIoT predictive maintenance. While we do not focus on predictive algorithms, we believe that the integration aspects are crucial for complex

maintenance as IIoT requests various services and supports beyond typical data analytics.

We use the BTS equipment maintenance in our prototype for the proof-of-concept. However, the framework is generic for different types of equipment in other domains, e.g., in hospitals. Results from a field experiment with real-world companies would be interesting for our future work. A key issue is how to map from data analytics results to domain knowledge so that the right professionals can be invoked. This requires a strong collaboration between IoT Cloud analytics, domain knowledge and human provisioning services. They will be the subject of our future experiments. Various aspects related to critical analytics results and human tasks generation are based on domain knowledge so that we will focus on optimizing the mapping and generation of suitable tasks. Furthermore, we will extend our model with multi-company expert systems by enabling different types of function catalogs, functions and human service provisioning systems.

ACKNOWLEDGMENTS

We thank Danh Pham and BachPhu for fruitful discussion about equipment monitoring and for sharing data of BTSs. Khiem Ta implements some analytics of Apache Spark that we use for our proof-of-concept. Muhammad Candra supports the integration with RAHYMS.

REFERENCES

[1] J. Wang, L. Zhang, L. Duan, and R. X. Gao, "A new paradigm of cloud-based predictive maintenance for intelligent manufacturing," *J. Intell. Manuf.*, vol. 28, no. 5, pp. 1125–1137, Jun. 2017.

[2] R. Sipos, D. Fradkin, F. Moerchen, and Z. Wang, "Log-based predictive maintenance," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '14. New York, NY, USA: ACM, 2014, pp. 1867–1876.

[3] "Azure iot suite predictive maintenance now available," https://blogs.microsoft.com/iot/2015/12/01/azure-iot-suite-predictive-maintenance-now-available/, last access: 25 Mar 2018.

[4] T. Foxworth, "Using iot and machine learning for industrial predictive maintenance," https://www.losant.com/blog/using-iot-and-machine-learning-for-industrial-predictive-maintenance, August 2017, last access: 25 March, 2018.

[5] *Directions in Hybrid Intelligence: Complementing AI Systems with Human Intelligence*, March 2016. [Online]. Available: https://www.microsoft.com/en-us/research/publication/directions-hybrid-intelligence-complementing-ai-systems-human-intelligence/

[6] "Iiot is greatbut dont forget about the humans," https://www.redcom.com/iiot-dont-forget-about-humans/, 16 Jan 2017.

[7] D. Haas, J. Ansel, L. Gu, and A. Marcus, "Argonaut: Macrotask crowdsourcing for complex data processing," *Proc. VLDB Endow.*, vol. 8, no. 12, pp. 1642–1653, Aug. 2015.

[8] V. Conotter, D.-T. Dang-Nguyen, M. Riegler, G. Boato, and M. Larson, "A crowdsourced data set of edited images online," in *Proceedings of the 2014 International ACM Workshop on Crowdsourcing for Multimedia*, ser. CrowdMM '14. New York, NY, USA: ACM, 2014, pp. 49–52. [Online]. Available: http://doi.acm.org/10.1145/2660114.2660120

[9] D. Haas, J. Wang, E. Wu, and M. J. Franklin, "Clamshell: Speeding up crowds for low-latency data labeling," *PVLDB*, vol. 9, no. 4, pp. 372–383, 2015.

[10] A. Karkouch, H. Mousannif, H. A. Moatassime, and T. Noel, "Data quality in internet of things: A state-of-the-art survey," *Journal of Network and Computer Applications*, vol. 73, pp. 57 – 81, 2016.

[11] P. Barnaghi, M. Bermudez-Edo, and R. Tönjes, "Challenges for quality of data in smart cities," *J. Data and Information Quality*, vol. 6, no. 2-3, pp. 6:1–6:4, Jun. 2015.

[12] H.-L. Truong and M. Halper, "Characterizing incidents in cloud-based iot data analytics," in *The 42nd IEEE International Conference on Computers, Software & Applications*, Tokyo, Japan, July 23-27.

[13] F. Rosenthal, C. Groba, A. Gossling, and S. Cech, "Architecture of a predictive maintenance framework," in *2007 6th International Conference on Computer Information Systems and Industrial Management Applications(CISIM)*, vol. 00, 06 2007, pp. 59–64.

[14] N. Savage, "Going serverless," *Commun. ACM*, vol. 61, no. 2, pp. 15–16, Jan. 2018. [Online]. Available: http://doi.acm.org/10.1145/3171583

[15] M. Yan, P. Castro, P. Cheng, and V. Ishakian, "Building a chatbot with serverless computing," in *Proceedings of the 1st International Workshop on Mashups of Things and APIs*, ser. MOTA '16. New York, NY, USA: ACM, 2016, pp. 5:1–5:4.

[16] S. Basu Roy, I. Lykourentzou, S. Thirumuruganathan, S. Amer-Yahia, and G. Das, "Task assignment optimization in knowledge-intensive crowdsourcing," *The VLDB Journal*, vol. 24, no. 4, pp. 467–491, Aug. 2015. [Online]. Available: http://dx.doi.org/10.1007/s00778-015-0385-2

[17] P. Mavridis, D. Gross-Amblard, and Z. Miklós, "Using hierarchical skills for optimized task assignment in knowledge-intensive crowdsourcing," in *Proceedings of the 25th International Conference on World Wide Web*, ser. WWW '16. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2016, pp. 843–853.

[18] J. Goncalves, M. Feldman, S. Hu, V. Kostakos, and A. Bernstein, "Task routing and assignment in crowdsourcing based on cognitive abilities," in *Proceedings of the 26th International Conference on World Wide Web Companion*, ser. WWW '17 Companion. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2017, pp. 1023–1031.

[19] M. Z. C. Candra, H. L. Truong, and S. Dustdar, "Provisioning quality-aware social compute units in the cloud," in *Service-Oriented Computing - 11th International Conference, ICSOC 2013, Berlin, Germany, December 2-5, 2013, Proceedings*, ser. Lecture Notes in Computer Science, S. Basu, C. Pautasso, L. Zhang, and X. Fu, Eds., vol. 8274. Springer, 2013, pp. 313–327.

[20] "OpsGenie," https://docs.opsgenie.com/, last access: 27 September 2018.

[21] M. C. Garcia, M. A. Sanz-Bobi, and J. del Pico, "Simap: Intelligent system for predictive maintenance application to the health condition monitoring of a windturbine gearbox," *Comput. Ind.*, vol. 57, no. 6, pp. 552–568, Aug. 2006.

[22] S. Windmann, A. Maier, O. Niggemann, C. Frey, A. Bernardi, Y. Gu, H. Pfrommer, T. Steckel, M. Krger, and R. Kraus, "Big data analysis of manufacturing processes," *Journal of Physics: Conference Series*, vol. 659, no. 1, p. 012055, 2015.

[23] S. Aromaa, A. Väätänen, I. Aaltonen, and T. Heimonen, "A model for gathering and sharing knowledge in maintenance work," in *Proceedings of the European Conference on Cognitive Ergonomics 2015*, ser. ECCE '15. New York, NY, USA: ACM, 2007, pp. 28:1–28:8.

[24] N. I. Badler, C. A. Erignac, and Y. Liu, "Virtual humans for validating maintenance procedures," *Commun. ACM*, vol. 45, no. 7, pp. 56–63, Jul. 2002. [Online]. Available: http://doi.acm.org/10.1145/514236.514264

[25] G. Shah and A. Tiwari, "Anomaly detection in iiot: A case study using machine learning," in *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data*, ser. CoDS-COMAD '18. New York, NY, USA: ACM, 2018, pp. 295–300.

[26] Y. Xu, Y. Sun, J. Wan, X. Liu, and Z. Song, "Industrial big data for fault diagnosis: Taxonomy, review, and applications," *IEEE Access*, vol. 5, pp. 17 368–17 380, 2017.

[27] E. H. Chi, "Technical perspective: Humans and computers working together on hard tasks," *Commun. ACM*, vol. 60, no. 9, pp. 92–92, Aug. 2017. [Online]. Available: http://doi.acm.org/10.1145/3068614

[28] G. Li, "Human-in-the-loop data integration," *Proc. VLDB Endow.*, vol. 10, no. 12, pp. 2006–2017, Aug. 2017.

[29] P. Minder and A. Bernstein, "Crowdlang: A programming language for the systematic exploration of human computation systems," in *Proceedings of the 4th International Conference on Social Informatics*, ser. SocInfo'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 124–137.

[30] D. W. Barowy, E. D. Berger, D. G. Goldstein, and S. Suri, "Voxpl: Programming with the wisdom of the crowd," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, ser. CHI '17. New York, NY, USA: ACM, 2017, pp. 2347–2358.