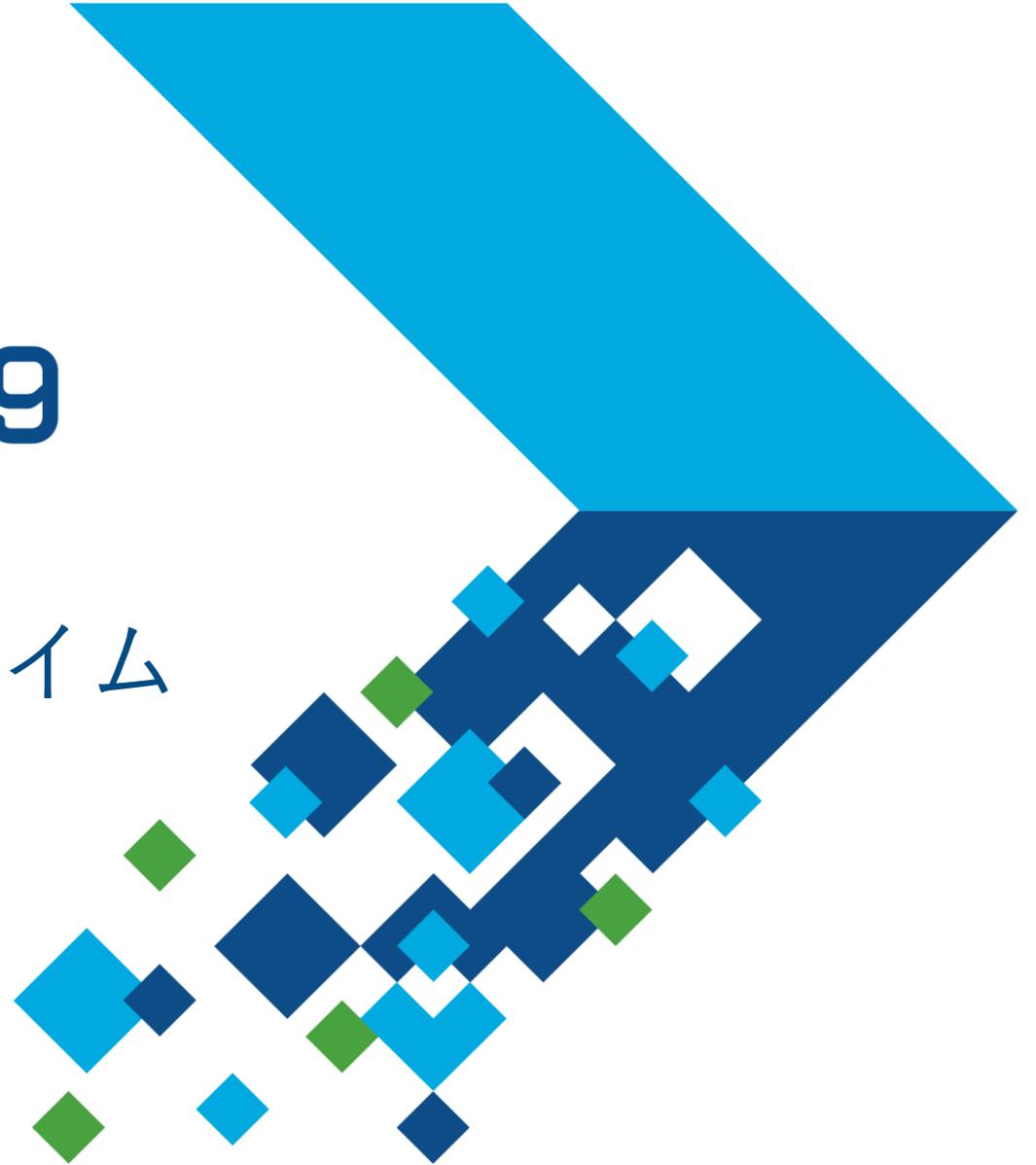


MATLAB EXPO 2019

モノづくりにおけるリアルタイム
の意思決定にAIを展開

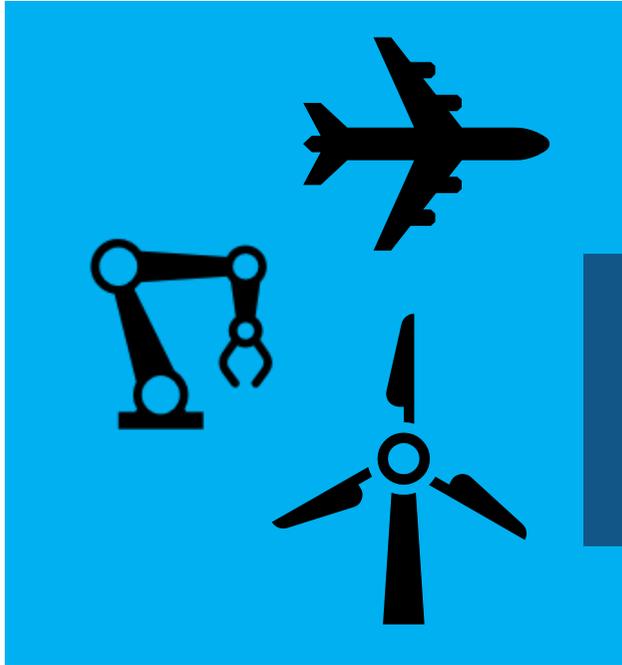
MathWorks® Japan
齊藤 甲次郎



アジェンダ

- はじめに
- 予知保全システム開発事例
 - 概要
 - 予知保全のモデル化
 - 予知保全モデルのシステム化
 - 予知保全システムの利用
- まとめ

リアルタイム解析が求められるワケ



予知保全

運用の効率向上

計画外のダウンタイムを削減

リアルタイム解析を必要とする
アプリケーションが増えています

ジェットエンジン: ~800 TB /day

タービン: ~2 TB /day

医療機器

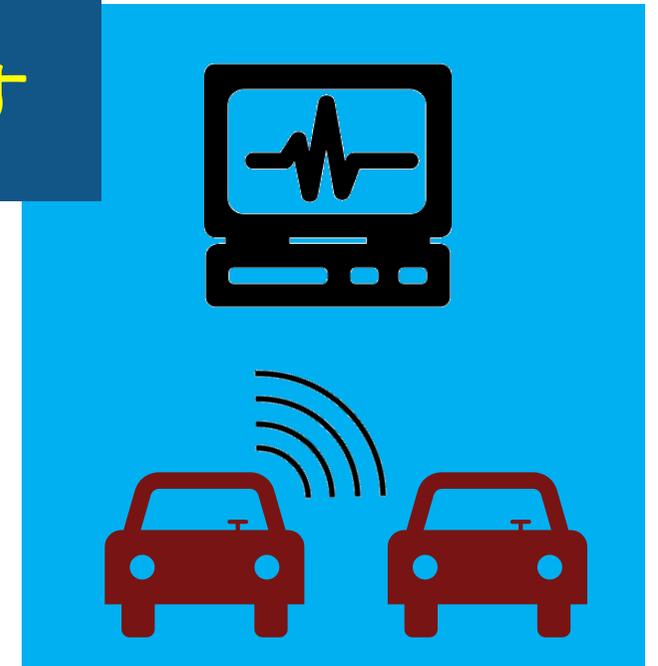
患者の安全

より良い治療効果

コネクテッドカー

安全、保守

先進運転

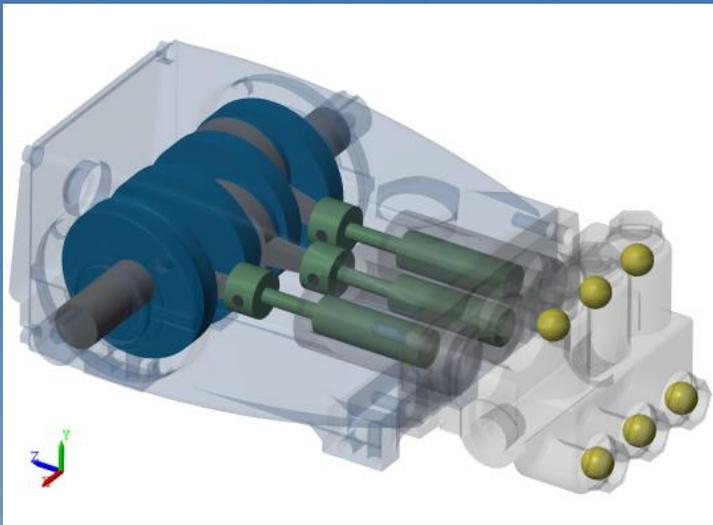


車: ~25 GB / hour



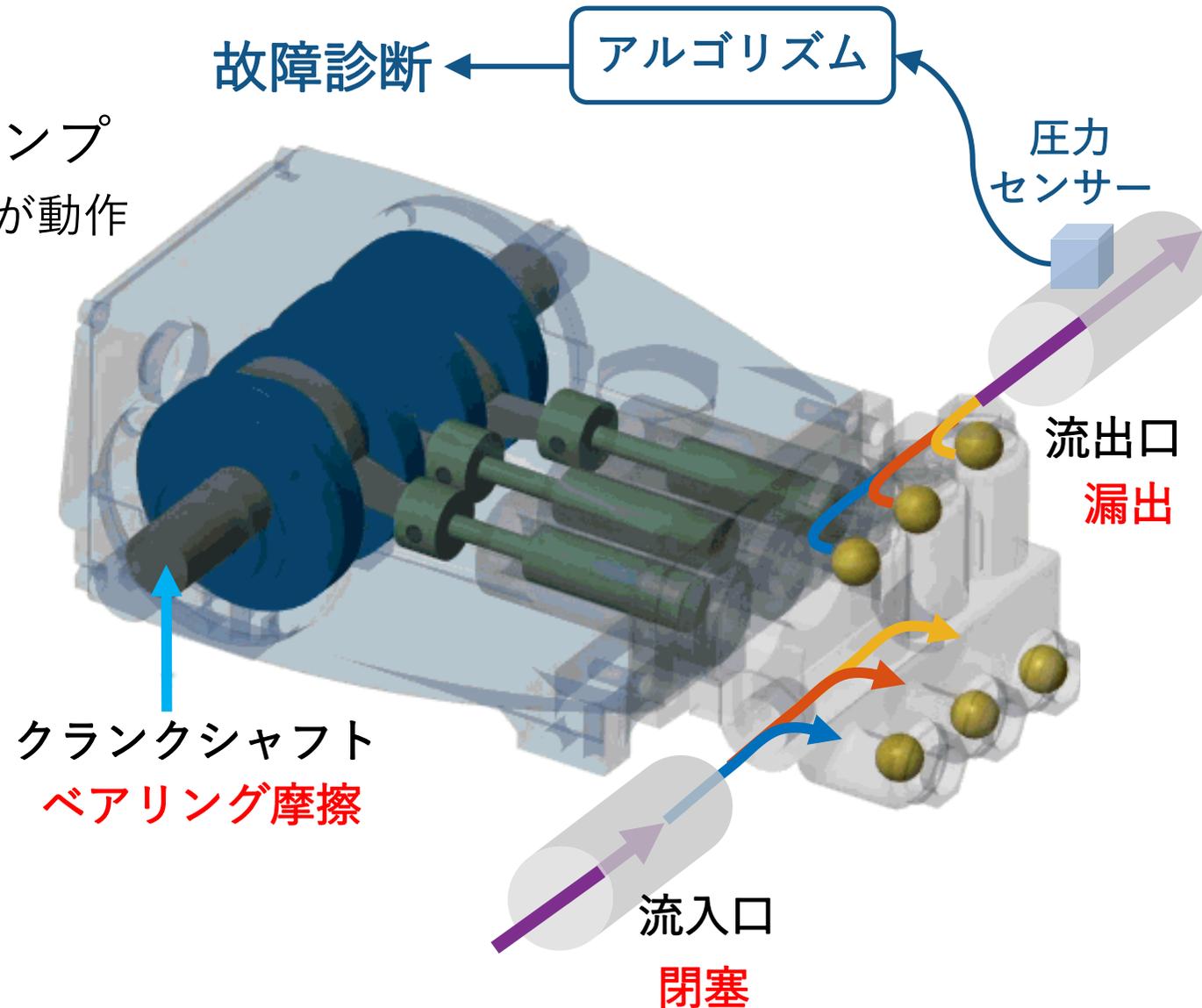
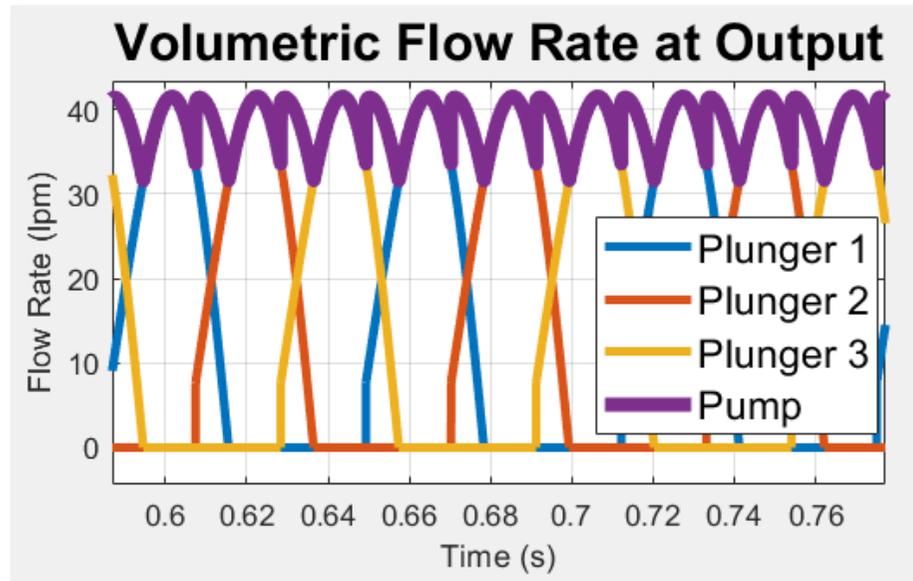
予知保全システム開発事例

～概要～



対象のポンプ

- 3つのプランジャーが動作するポンプ
 - 120° ずつの位相差でプランジャーが動作
 - 1つのチャンバーが常に排出
 - 3種類の故障



予知保全プロジェクトの担当者



プロセスエンジニア

予知保全の解析モデル
を開発



システムアーキテクト

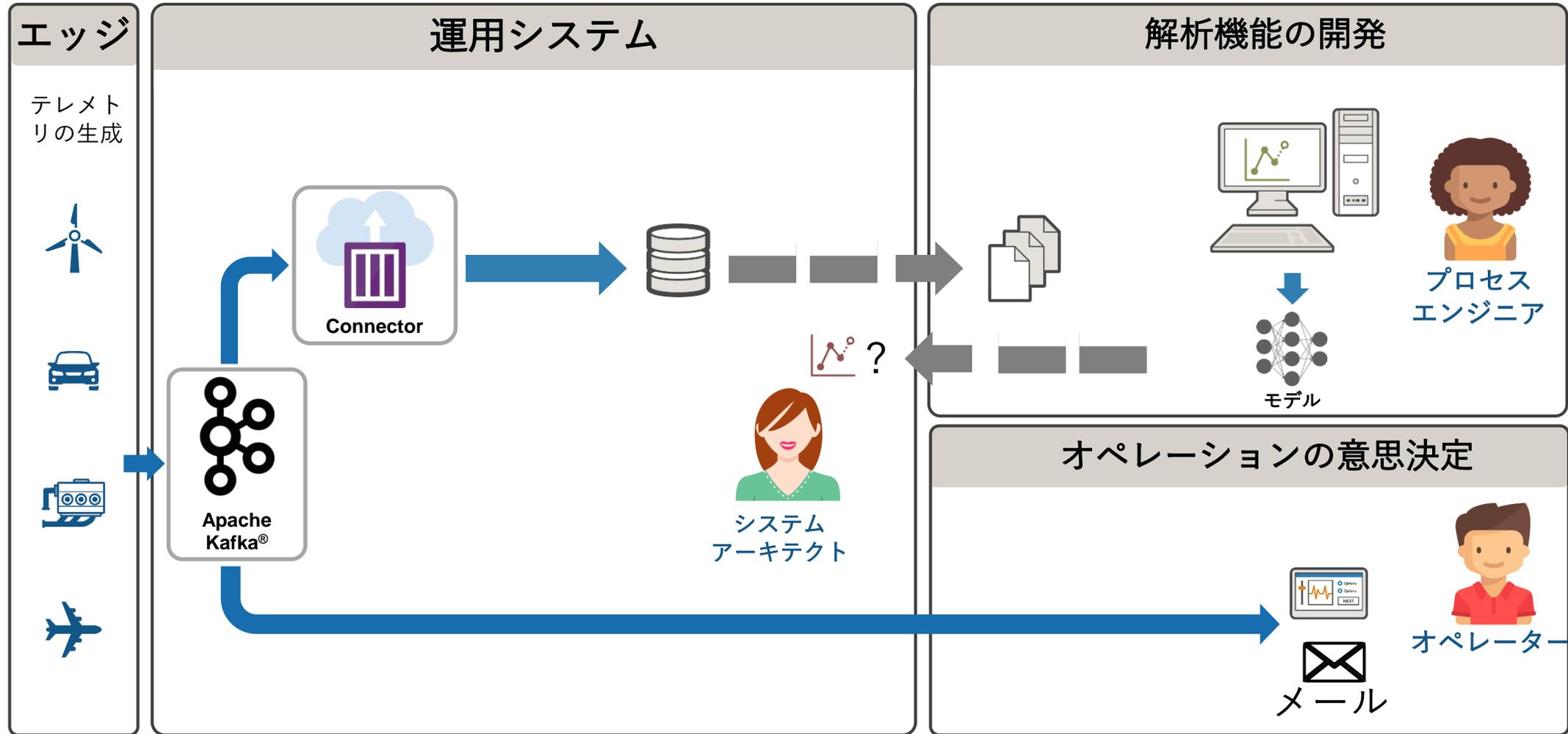
予知保全のシステム
を開発



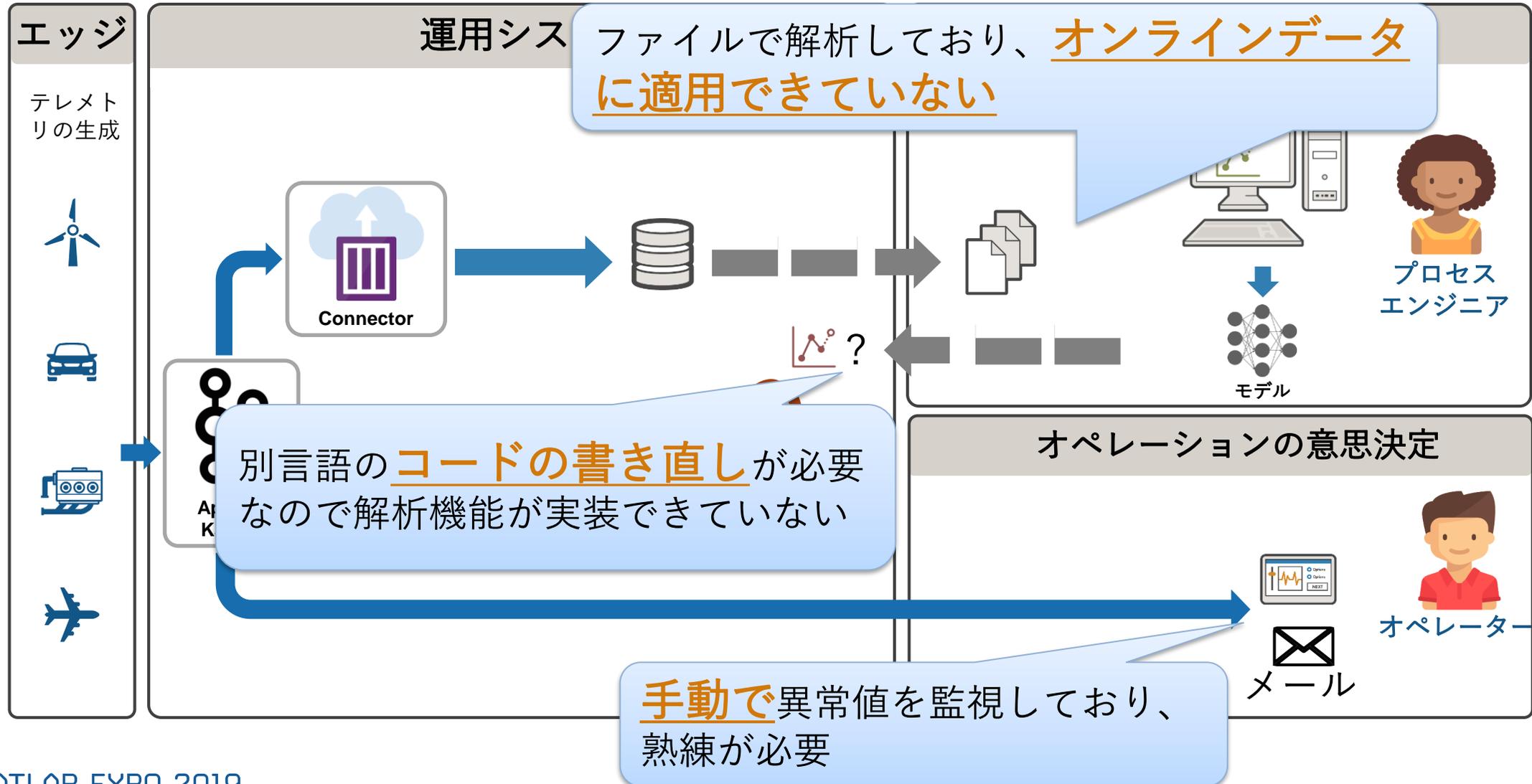
プラントオペレーター

システムを利用して
オペレーションの意思決定

現状の予知保全システムと課題



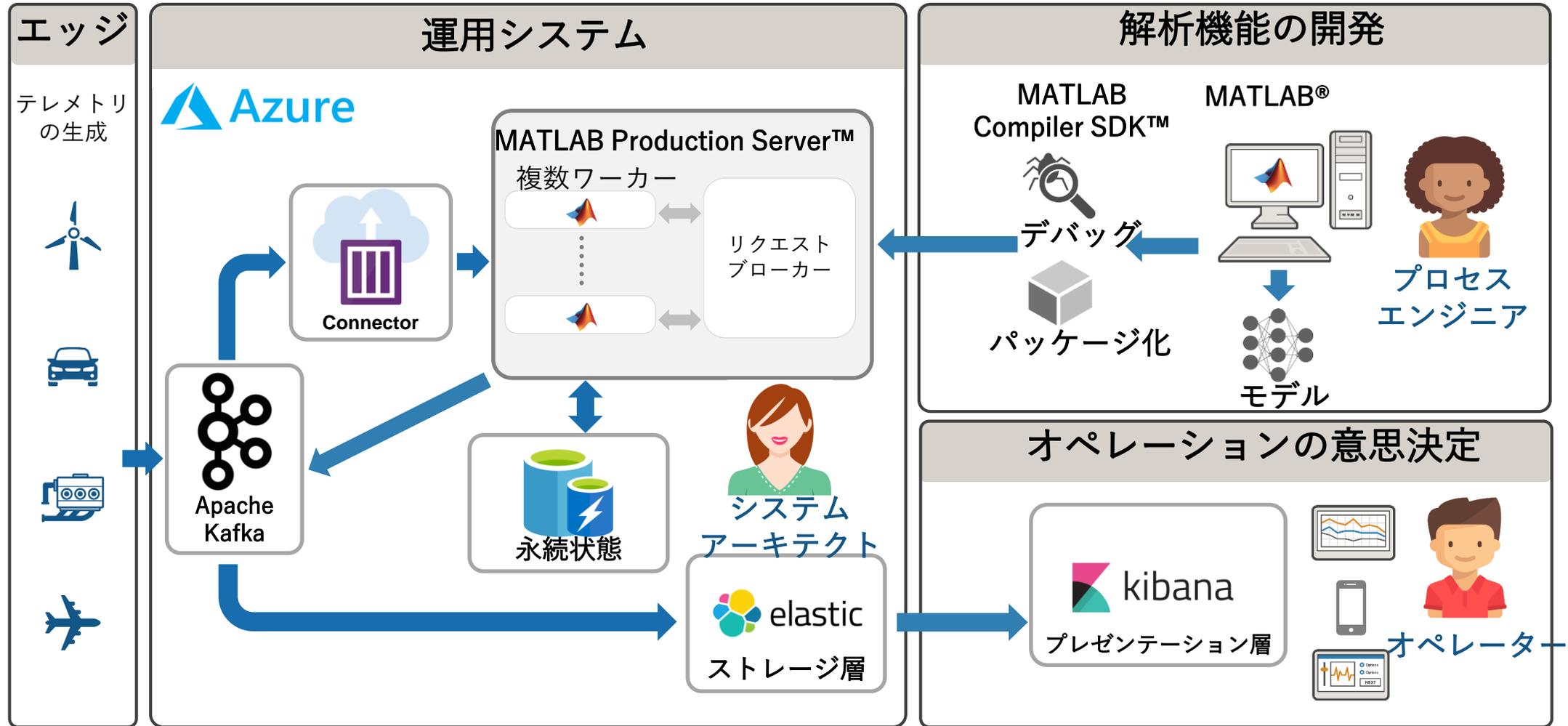
現状の予知保全システムと課題



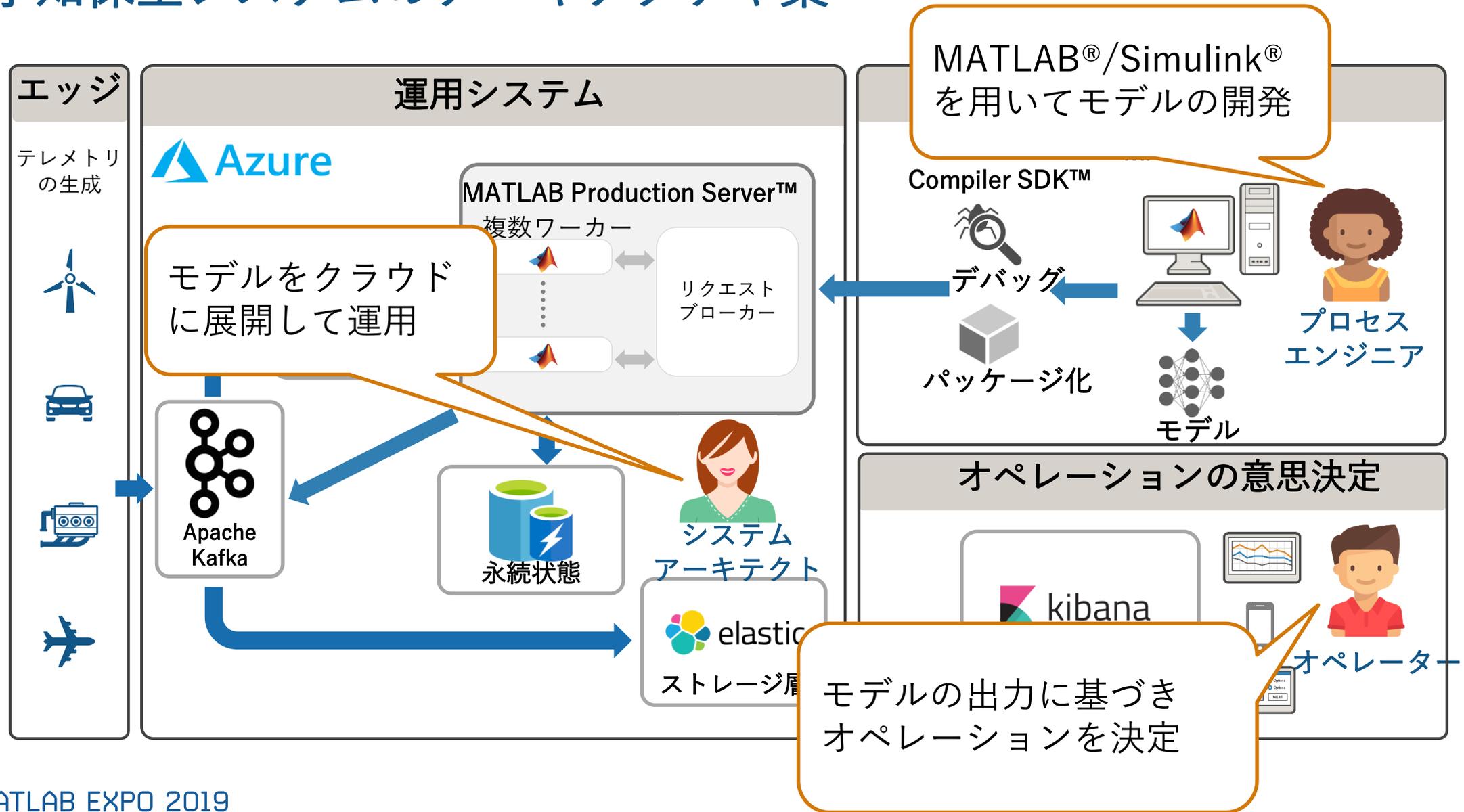
予知保全システム開発のプロジェクト概要

| 項目 | 概要 |
|----|---|
| 期間 | 1ヶ月 |
| 内容 | <ul style="list-style-type: none"> • 予知保全のアルゴリズム開発(モデル化) • モデルのシステム化 • ストリーミング処理 |
| 要求 | <p>オペレーションからの要求</p> <ul style="list-style-type: none"> • 全ポンプの流量、圧力、電流を監視してオペレーションの状態を常時把握できること • パラメータが範囲外にドリフトした際にすぐにリアクションができるようにすること • 全ポンプの故障までの寿命を常時推定し、保守や交換をスケジューリングできること <p>システムアーキテクトからの要求</p> <ul style="list-style-type: none"> • ITの予算が限られているが、投資する前にソリューションがうまくいくか把握できること |

予知保全システムのアーキテクチャ案



予知保全システムのアーキテクチャ案



予知保全システム開発のアプローチ



予知保全システム開発事例

～予知保全のモデル化～



1

データへのアクセスと探索

プロセス
エンジニア

ヒストリカルデータの利用

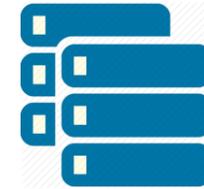
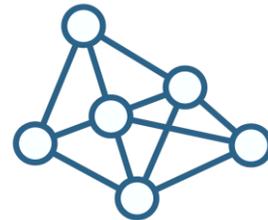
① バッチ処理：ヒストリカルデータでモデルの開発とテスト

ヒストリカルデータ

モデルの学習

スケールアップ

予測

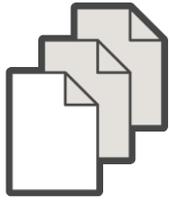




1

データへのアクセスと探索

プロセス
エンジニア

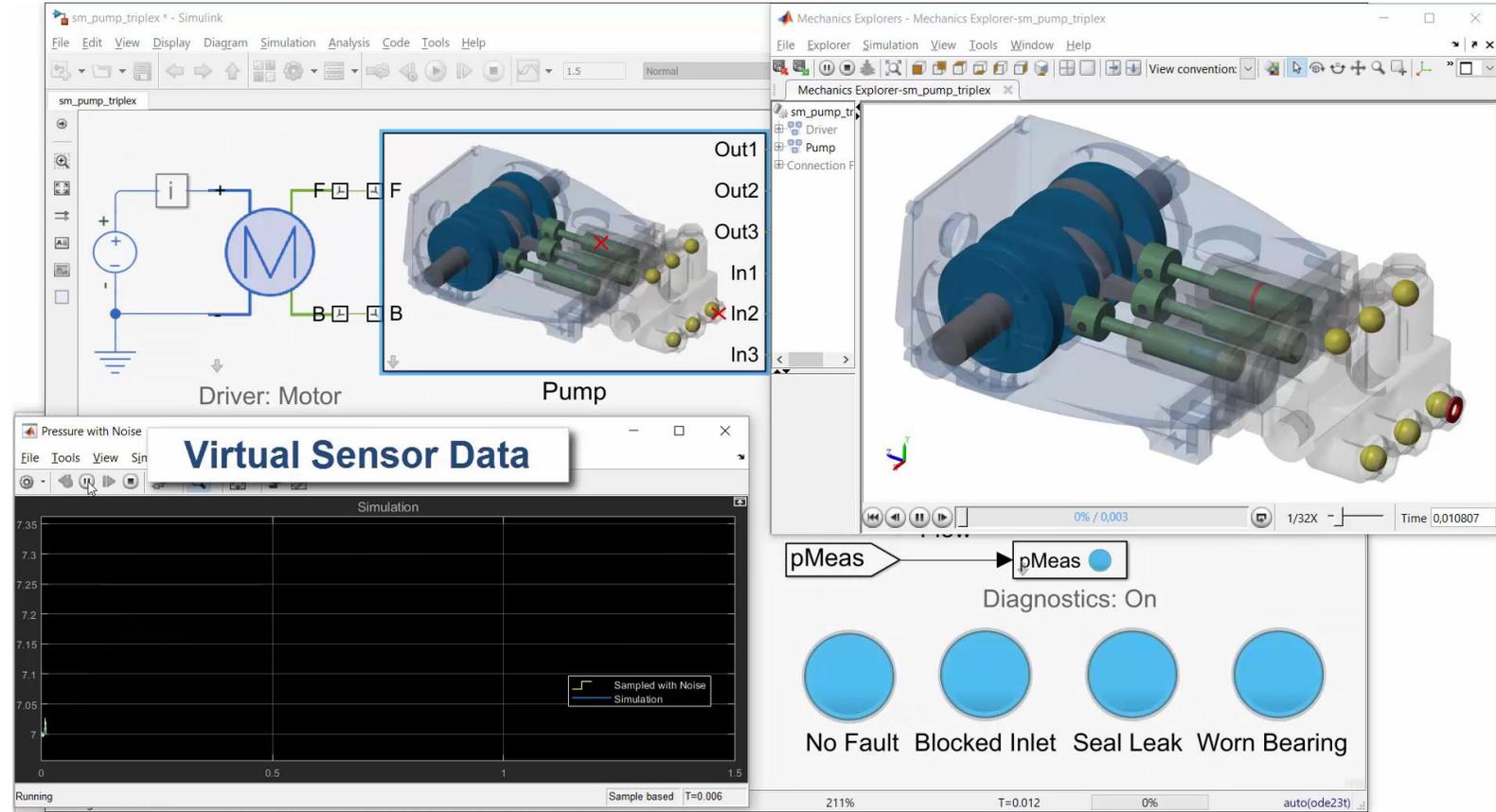


ファイル
(正常データ)



シミュレーション
(異常データ)

故障レベルの特定にセンサデータを利用



Simulinkモデルで故障をシミュレーション
(故障データがあまり無く、実際に装置を故障
させるのはコストが掛かりすぎるため)



2

データの前処理

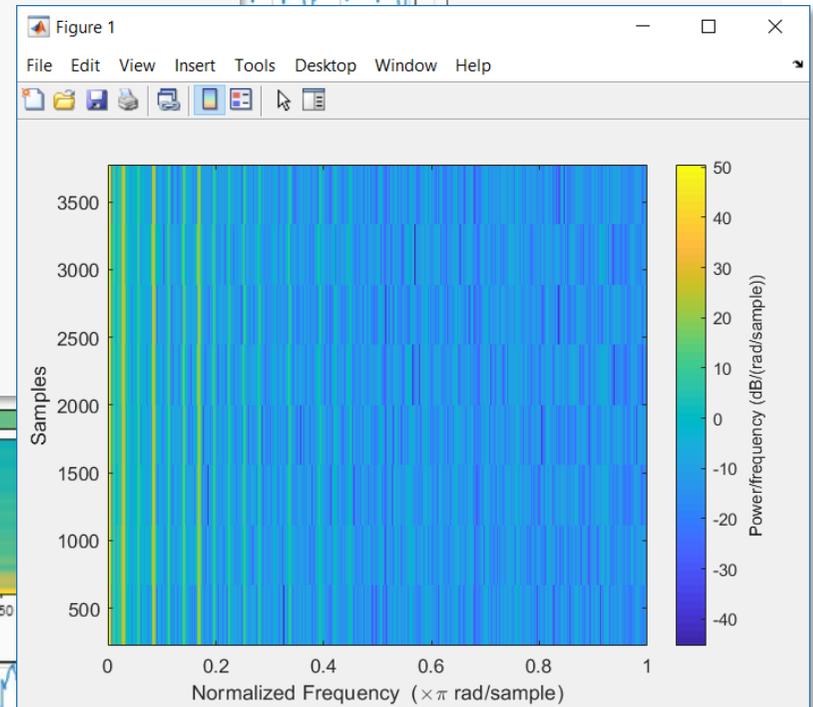
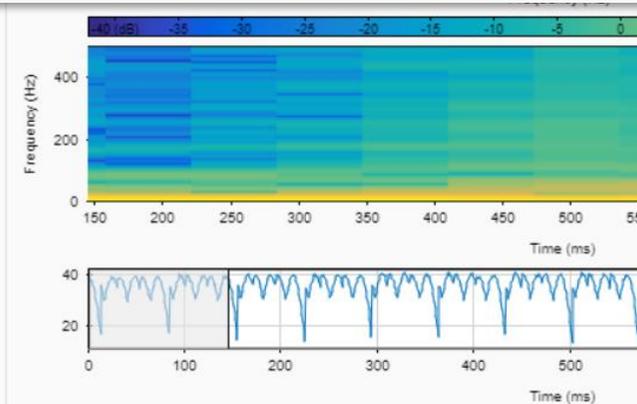
信号データの前処理

プロセス
エンジニア

Signal processing

```
[Spectrum, Frequencies] = pspectrum(data.Flow);
[pLow, pHigh] = bounds(Spectrum);
fPeak = Frequencies(Spectrum==pHigh);
qPeak2Peak = peak2peak(data.Flow);
qCrest = peak2rms(data.Flow);
qRMS = rms(data.Flow);
qMAD = mad(data.Flow);
```

| NAME | SIZE | CLASS |
|--------------|--------|-----------|
| allfaults | 1000×3 | timetable |
| bearingPump | 1000×3 | timetable |
| blockedPu... | 1000×3 | timetable |
| healthyPump | 1000×3 | timetable |
| leakingPump | 1000×3 | timetable |



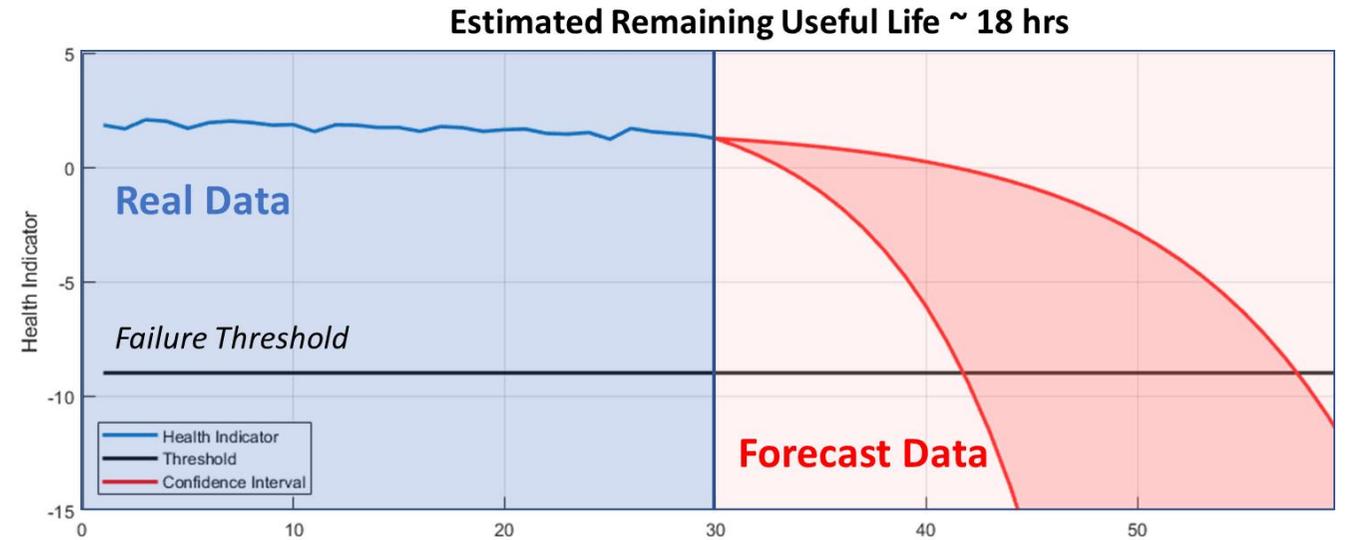
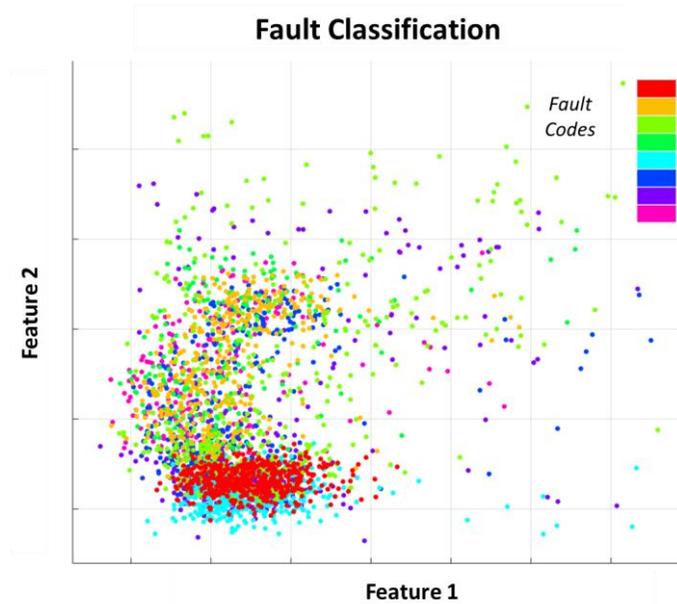


3

予測モデルの開発

プロセス
エンジニア

MATLABでの予測モデルの開発



故障の種類
→ 分類



オペレーターに
必要な情報

故障までの予測時間
→ 回帰

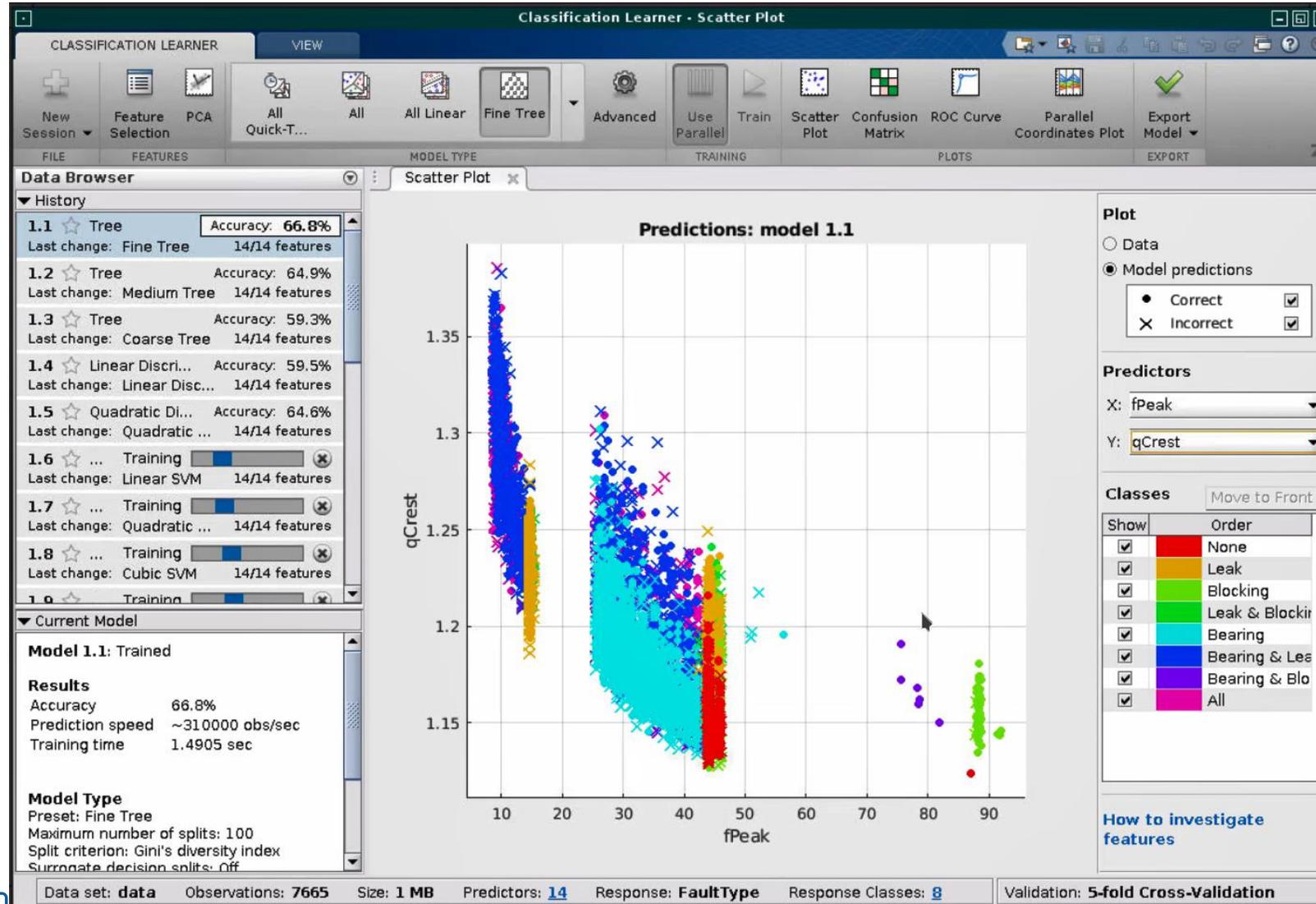


3

予測モデルの開発

プロセス
エンジニア

機械学習モデル(分類)の開発



- 分類ラベル
- Blocking
 - Leaking
 - Bearing
 - 上記組合せ

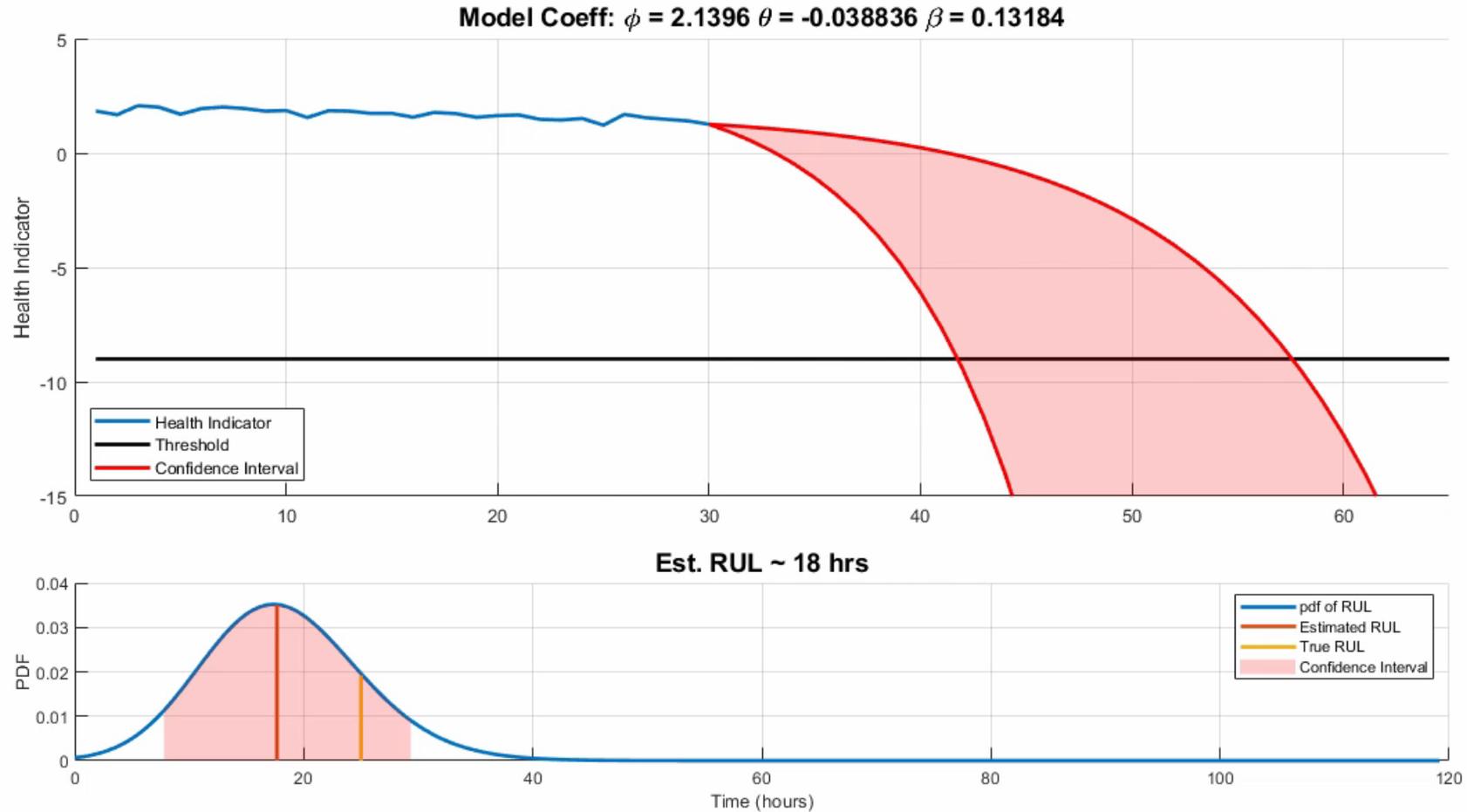


3

予測モデルの開発

プロセス
エンジニア

故障までの時間を予測(回帰)



$$S(t) = \phi + \theta(t) e^{(\beta(t)t + \epsilon(t) - \frac{\sigma}{2})}$$



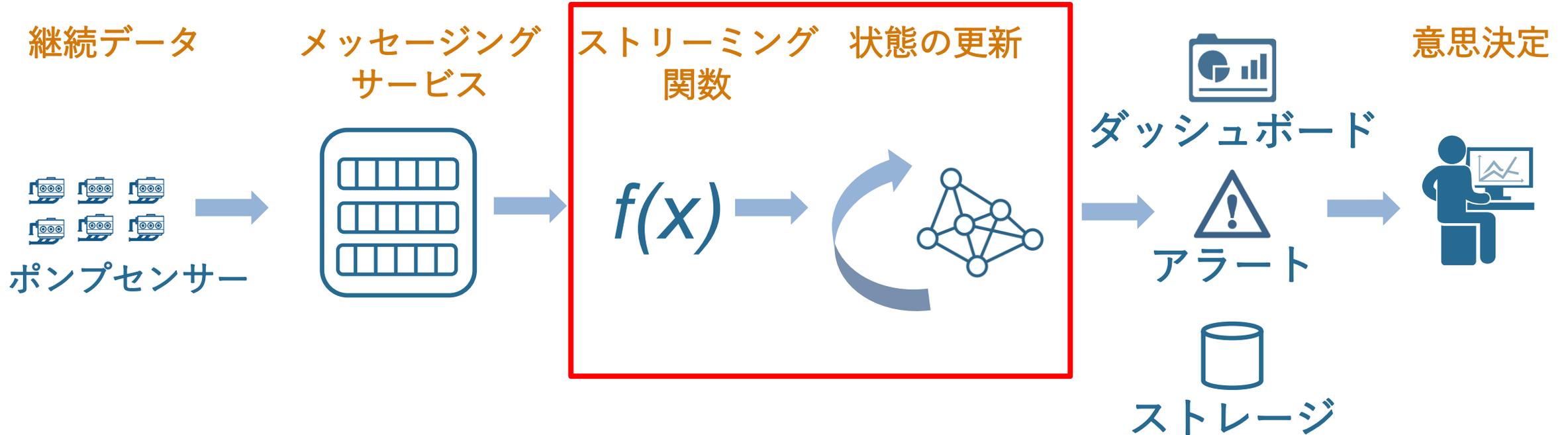
4

システムへの統合

ストリーム処理関数を開発

プロセス
エンジニア

②ストリーム処理: リアルタイムでセンサーデータにモデルを適用





4

システムへの統合

ストリーム処理関数を開発

プロセス
エンジニア

Streaming Function

```
function new_state = streamingFunction(data, old_state)
```

Preprocess signals

```
[data, features] = preprocessData(data);
```

Predict faults

```
[Leak, Blocking, Bearing] = predictFaultValues(features);  
FaultType = predictFault(features);  
[RUL, Model] = predictUpdateRUL(data.Timestamp, data.Flow, 500);
```

Update state

```
new_state = updateState(data, old_state);
```

Write results

```
writeResults(Leak, Blocking, Bearing, FaultType, RUL, Model)  
end
```

データが送られてきたら
全ての窓を処理

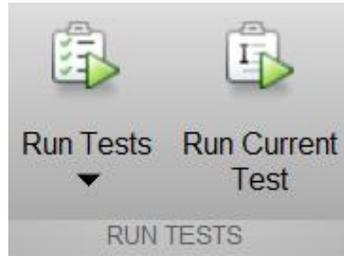
1つ前の状態

処理される現在のデータ窓



4

システムへの統合

プロセス
エンジニア

```
results = runtests('predictFaults_tests')
```

```
Running predictFaults_tests
....
Done predictFaults_tests
```

```
results =
1x4 TestResult array with properties:
```

```
Name
Passed
Failed
Incomplete
Duration
Details
```

```
Totals:
4 Passed, 0 Failed, 0 Incomplete.
0.01614 seconds testing time.
```

ストリーム処理関数のテスト

The screenshot shows the Production Server Compiler interface. The 'Test Client' button is highlighted with a red box. Below it, the 'TEST' tab is active, showing the 'Server Address' field with the value 'http://localhost:9910/streamingFunction'. The 'Server Requests' section is empty, and the 'Server Log' section shows the following output:

```
22 [Sat Nov 03 18:54:15 EDT 2018] binding to endpoint '[:,:]:9910'
23 [Sat Nov 03 18:54:15 EDT 2018] using '[:,:]:9910'
24 [Sat Nov 03 18:54:15 EDT 2018] bound to '[:,:]:9910'
25 [Sat Nov 03 18:54:15 EDT 2018] [SERVICE:date_clock] Starting
26 [Sat Nov 03 18:54:15 EDT 2018] [SERVICE:date_clock] Started successfully
27 [Sat Nov 03 18:54:15 EDT 2018] [SERVICE:http] [endpoint[:,:]:9910] Starting
28 [Sat Nov 03 18:54:15 EDT 2018] [SERVICE:http] [endpoint[:,:]:9910] Waiting for a connection on port 9910
29 [Sat Nov 03 18:54:15 EDT 2018] [SERVICE:http] [endpoint[:,:]:9910] Started successfully
30 [Sat Nov 03 18:54:15 EDT 2018] [SERVICE:http] [endpoint[:,:]:9910] Listening for connections
```

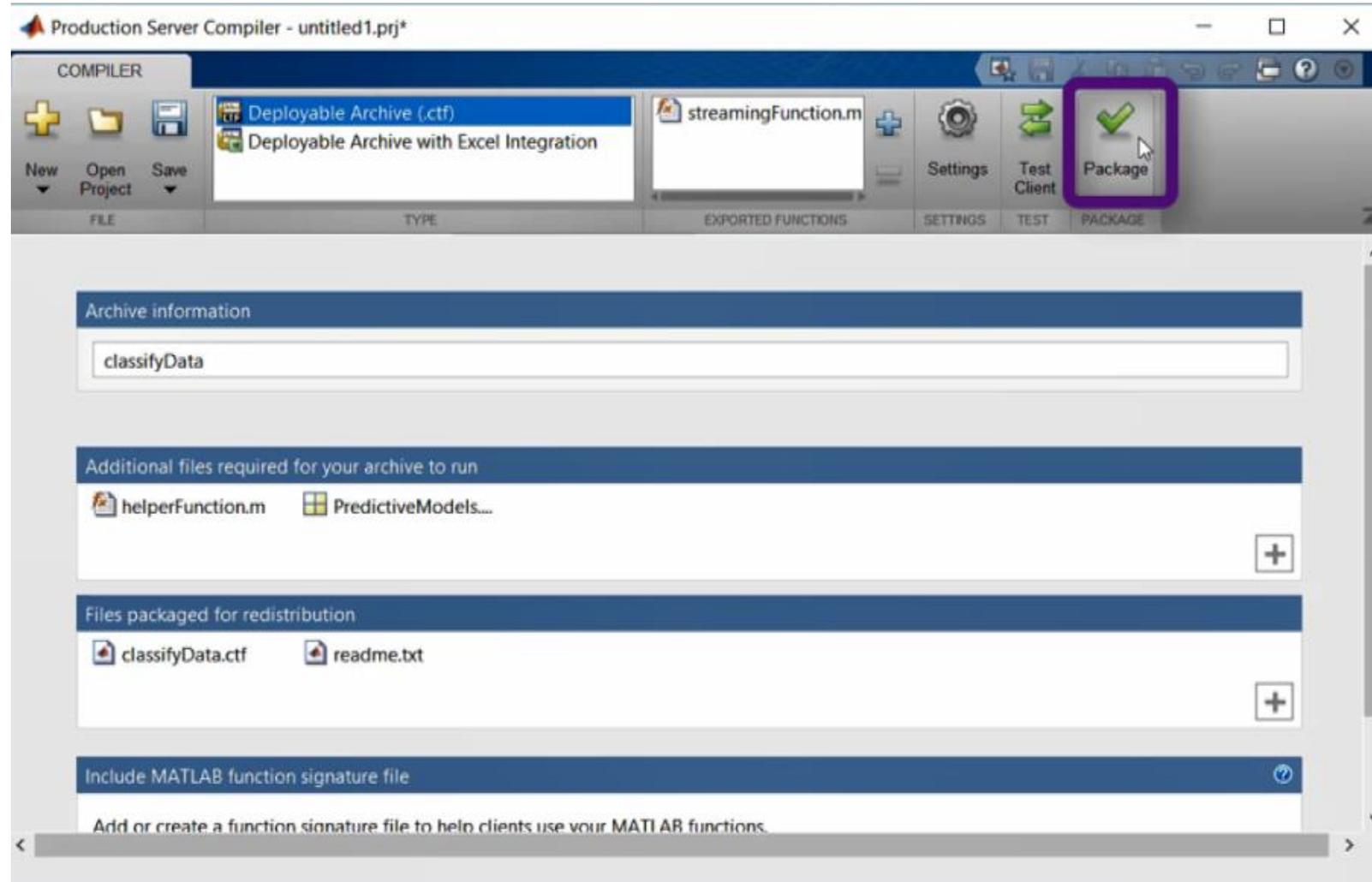


4

システムへの統合

プロセス
エンジニア

ストリーム処理関数のパッケージ化



予知保全システム開発事例

～予知保全モデルのシステム化～

4

システムへの統合

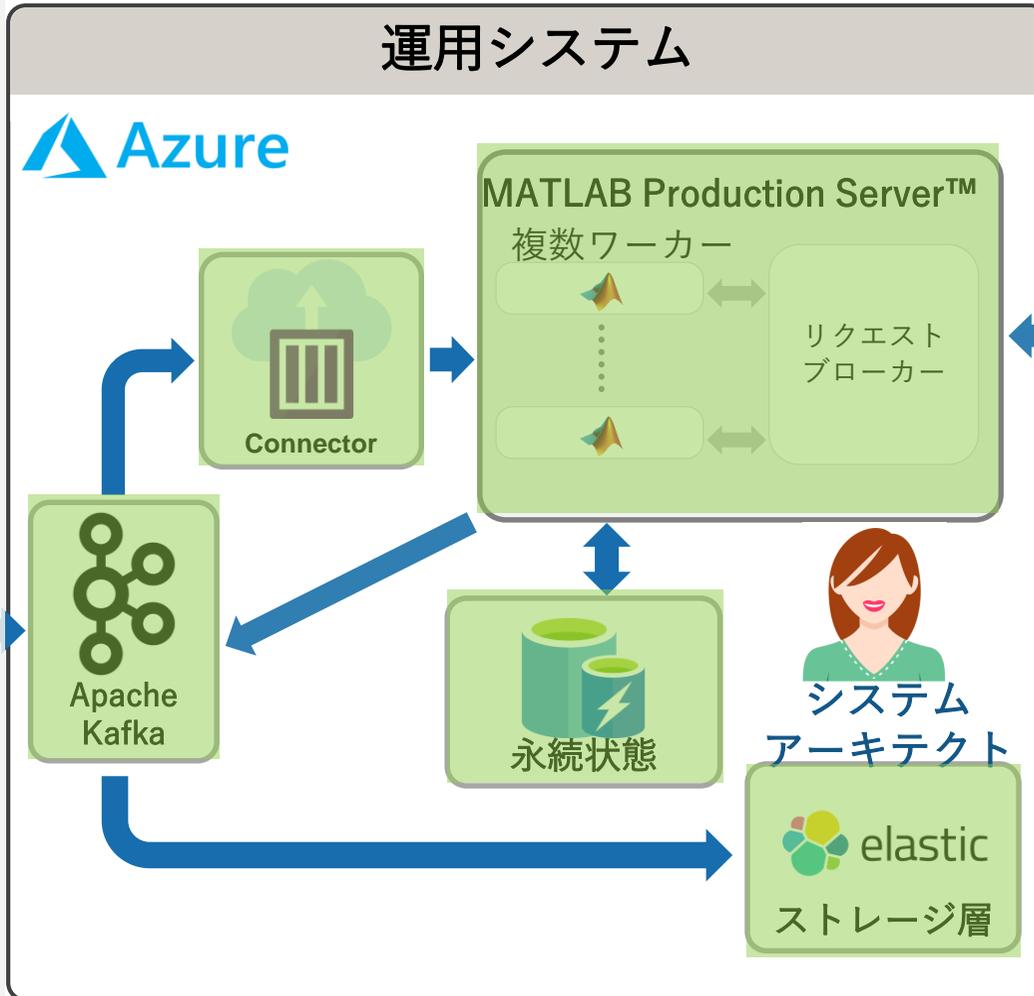
システムへの解析の統合



システム
アーキテクト

エッジ

テレメトリ
の生成



4

システムへの統合

Kafkaを用いたストリーミングデータの入出力

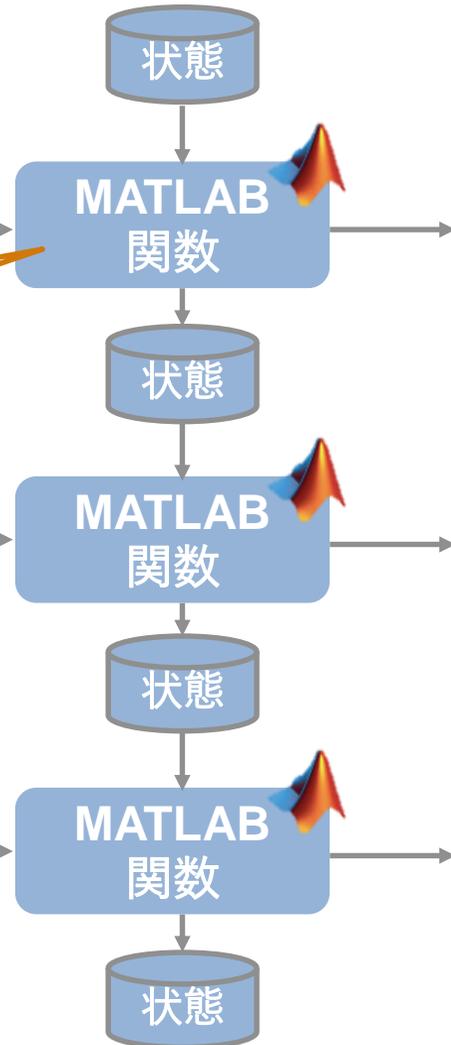


システム
アーキテクト

入カストリーム

| Event Time | Pump Id | Flow | Pressure | Current |
|------------|---------|------|----------|---------|
| 18:01:10 | Pump1 | 1975 | 100 | 110 |
| 18:10:30 | Pump3 | 2000 | 109 | 115 |
| 18:01:10 | ... | ... | ... | ... |
| 18:10:30 | ... | ... | ... | ... |
| 18:30:10 | Pump4 | 2000 | 100 | 110 |
| 18:35:20 | Pump4 | 1960 | 103 | 105 |
| 18:20:40 | Pump3 | 1970 | 112 | 104 |
| 18:39:30 | Pump4 | 2100 | 105 | 110 |
| 18:30:00 | Pump3 | 1980 | 110 | 113 |
| 18:30:50 | Pump3 | 2000 | 100 | 110 |
| ... | ... | ... | ... | ... |

バッチ処理で作成したモデル
をストリーム処理に利用



出カストリーム

| Time window | Pump Id | Bearing Friction |
|-------------|----------|----------------------------------|
| ... | ... | ... |
| 18:00:00 | 18:10:00 | Pump1: 5, Pump3: ..., Pump4: ... |
| 18:10:00 | 18:20:00 | Pump2: 7, Pump3: 3, Pump4: ... |
| 18:20:00 | 18:30:00 | Pump1: ..., Pump3: 4, Pump4: ... |
| 18:30:00 | 18:40:00 | Pump5: ..., Pump3: 5, Pump4: 8 |

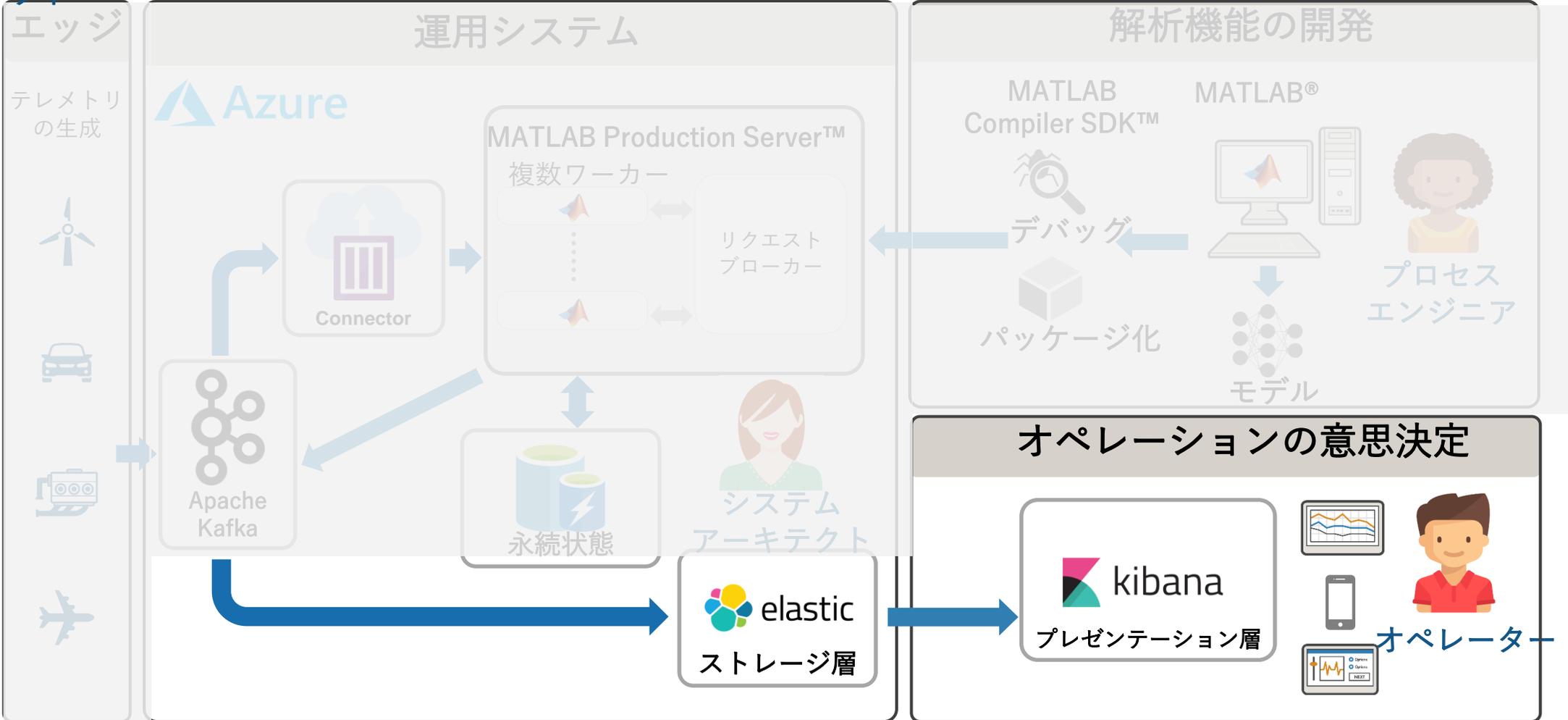
4

システムへの統合

ダッシュボードの作成



システム
アーキテクト



予知保全システム開発事例

～予知保全システムの利用～



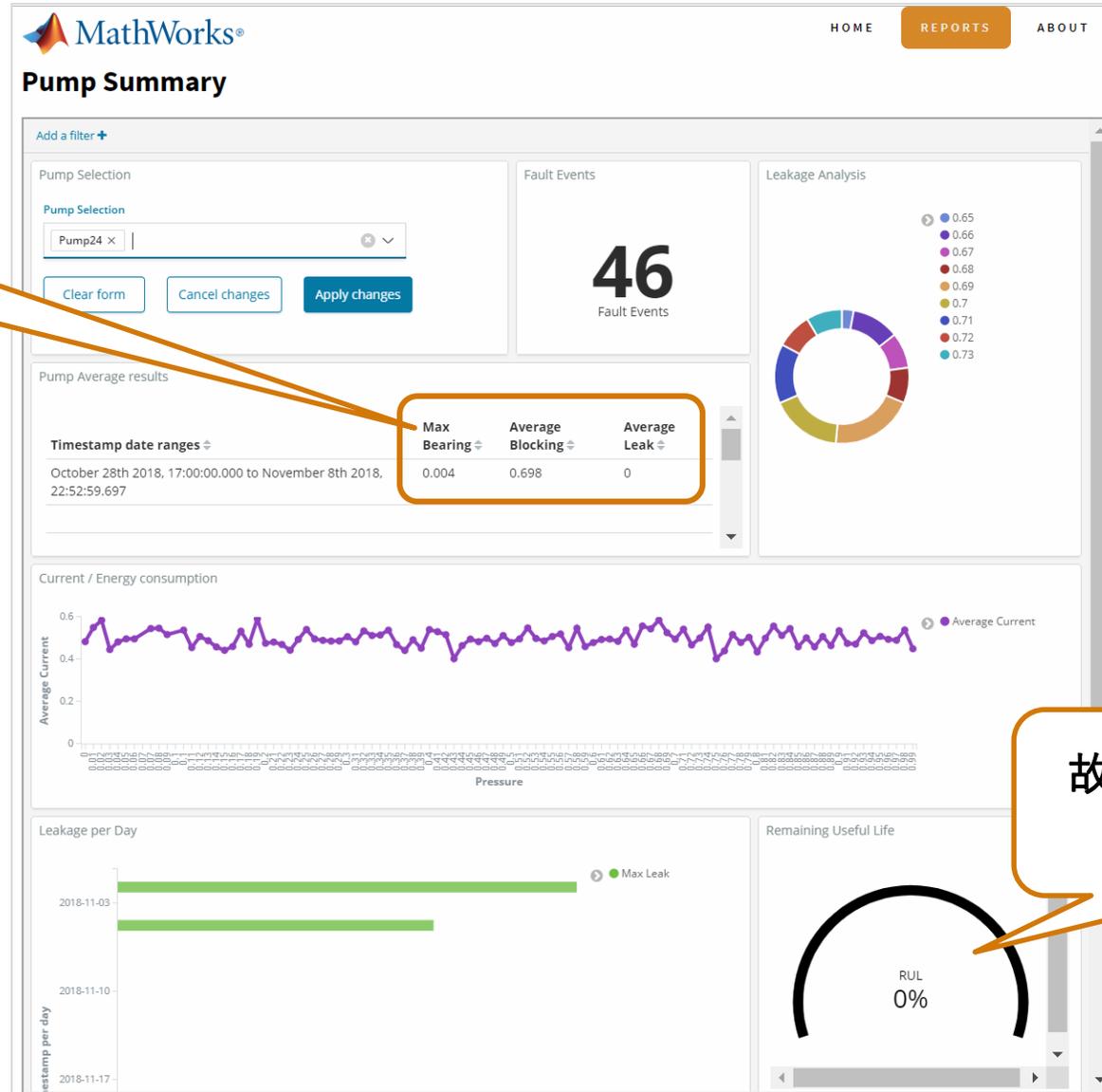
5

結果の可視化

ダッシュボードの利用

プラント
オペレーター

故障の種類を
予測



故障までの寿命の
予測

まとめ

Key Takeaways

- MATLABの**アプリを活用した**予知保全モデルの開発
- 実際に故障した際の機器のデータが無くても、
Simulinkを用いて**故障した場合のシミュレーションデータ**を作成
- プロトタイプ作成から、ストリーミングプラットフォームKafkaとクラウドを用いたシステム化までを**3週間**で実現



© 2019 The MathWorks, Inc. MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

参考：使用したToolbox

| 項目 | 使用した製品 |
|-----------------------|---|
| 予知保全のモデル化 | <ul style="list-style-type: none"> • MATLAB® • Predictive Maintenance Toolbox™ • Signal Processing Toolbox™ • Statistics and Machine Learning Toolbox™ |
| ポンプのシミュレーション データ作成 | <ul style="list-style-type: none"> • Simulink® • Simscape™ • Simscape™ Electrical™ • Simscape™ Fluids™ • Simscape™ Multibody™ • Predictive Maintenance Toolbox • Parallel Computing Toolbox™ |
| 予知保全モデルのシステム化 | <ul style="list-style-type: none"> • MATLAB® Compiler SDK™ • MATLAB Production Server™ |

参考：補足情報

- GitHub: MathWorks Reference Architectures

<https://github.com/mathworks-ref-arch>

- MATLABとIT システムの連携

<https://jp.mathworks.com/solutions/enterprise-it-systems.html>

- データサイエンス向けMATLAB

<https://jp.mathworks.com/solutions/data-science.html>



MATLAB[®] コードは、そのまま安全に展開してエンタープライズ IT システム、データソース、運用・制御技術と統合することができます。IT とエンジニアリング チームが手を組むことで、次のことが可能になります。

- オンプレミスかパブリッククラウド (AWS[®] や Microsoft[®] Azure[®] など) かを問わず、Windows[®] および Linux[®] 上で信頼性、安全性、拡張性の高い運用アプリケーションを実行できます。
- 業界標準のセキュリティメカニズムを利用して、認証、データへのアクセス権の付与、暗号化を行うことができます。
- Tableau[®]、TIBCO[®] Spotfire[®]、Power BI などの最新のアナリティクス システムを含め、既存のシステムやデータに直接統合できます。

「MATLAB Compiler を使ってスタンドアロンの運用プログラムを作成して自動で実行するだけで、最新の予測データを Horizon のアナリストに毎日提供できます。エンタープライズサーバーのセットアップは IT 部門が行いましたが、その後は IT 部門の手を借りることなく自分たちで簡単にプログラムを更新できます」

— Manuel Arancibia and Cedric KouamManuel Arancibia[®], Cedric Kouam[®], Horizon Wind Energy

