The Between Software Engineering and Artificial Intelligence

Challenges and an Agenda for MDE

Joost Noppen, BT Applied Research

1 December 2022

General. v 1. Joost Noppen.

Who am I and Who do I work for?

Joost Noppen
Chief Researcher Software
BetaLab, BT Applied Research
Adastral Park, Ipswich
johannes.noppen@bt.com

British Telecom
Broadband & Phones, right?

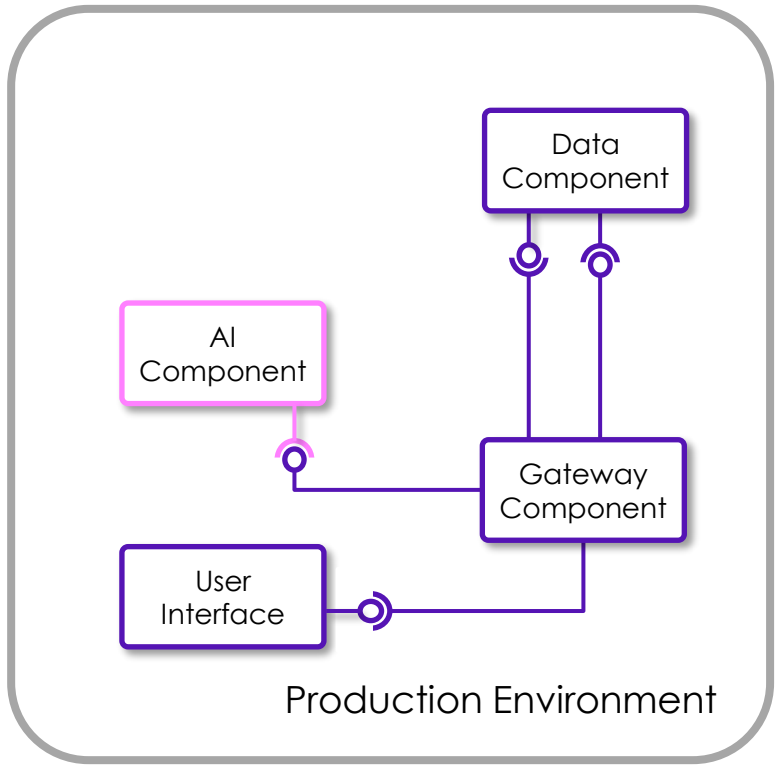
Well yes, but also:

Virtualised networks, multi-media, content creation, media delivery, mobile portals, big data science, virtual reality, remote surgery, digital avatars, resource allocation, car maintenance, …

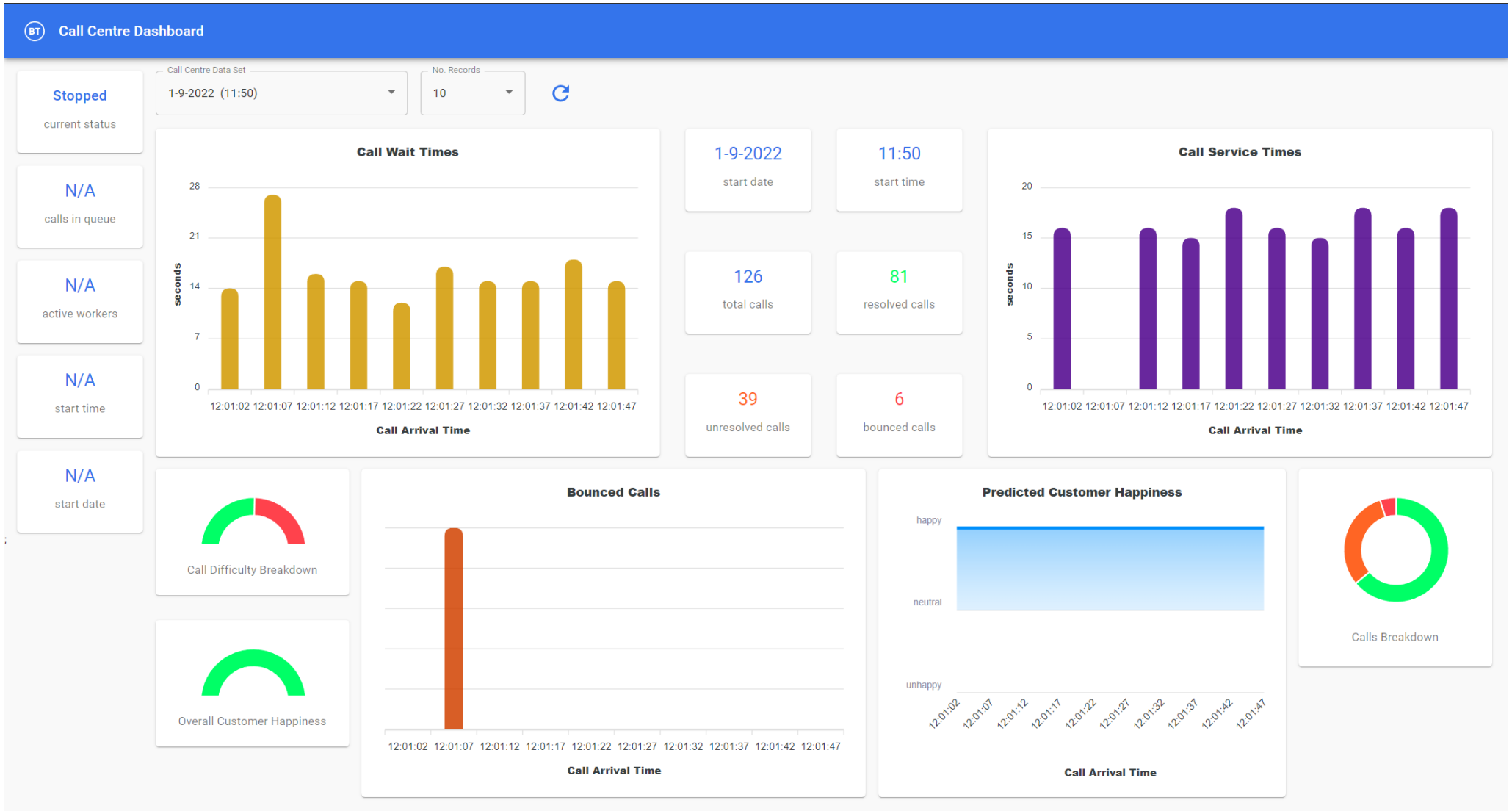
In short, a massive company with at its core two defining features:

Software & Artificial Intelligence

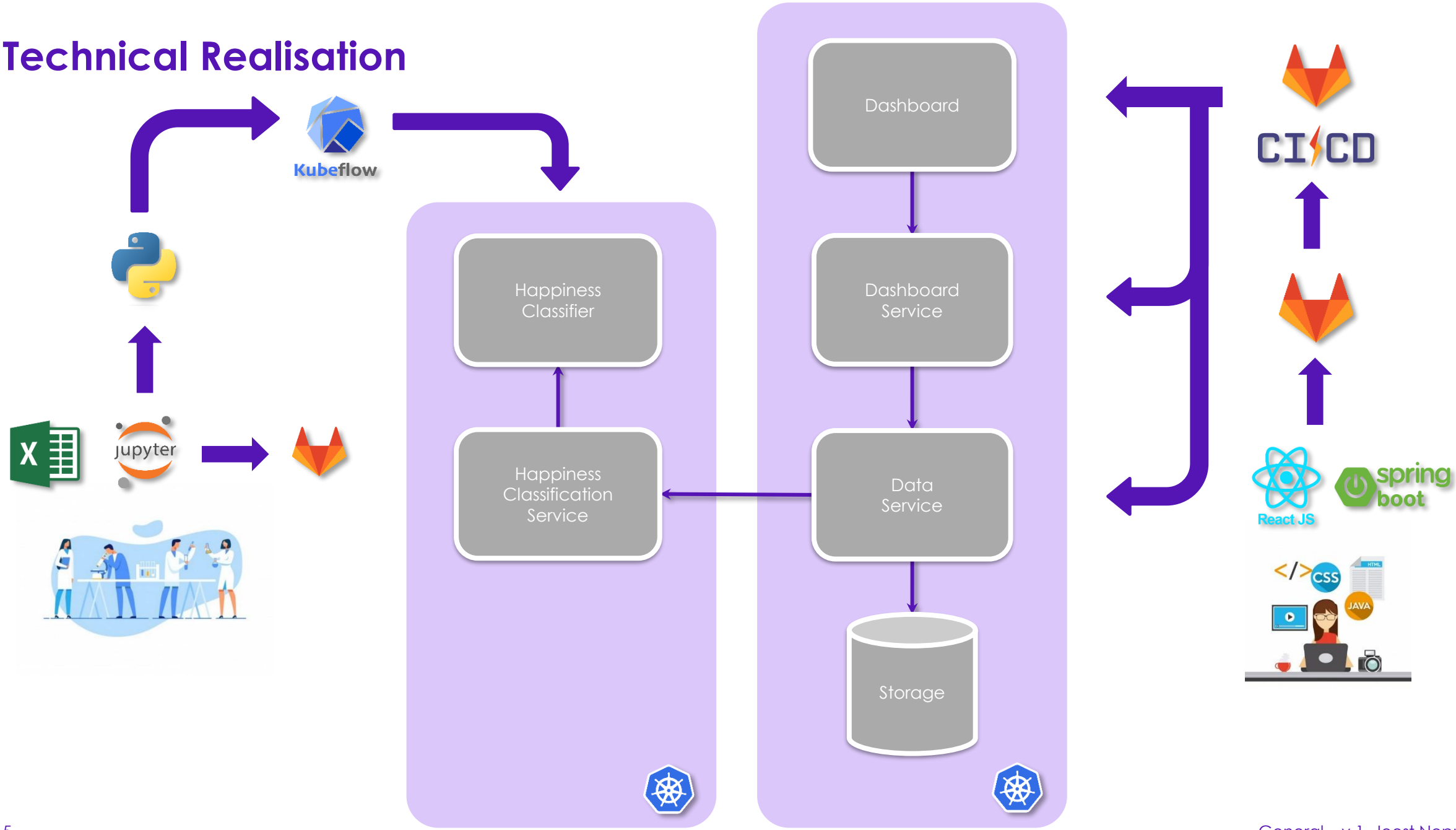
Software Engineering and AI – The Same Goal but Worlds Apart



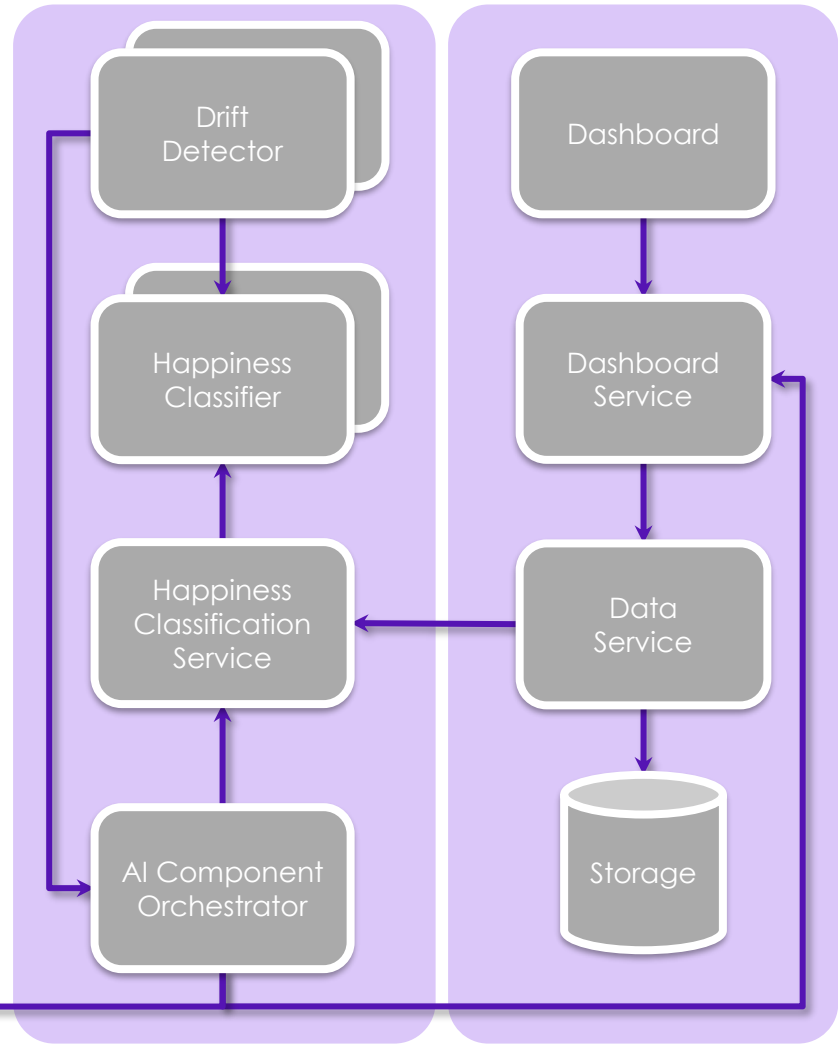
Call Centre Dashboard

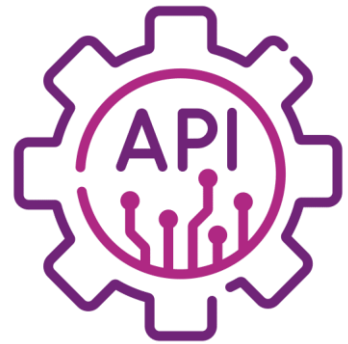


Technical Realisation



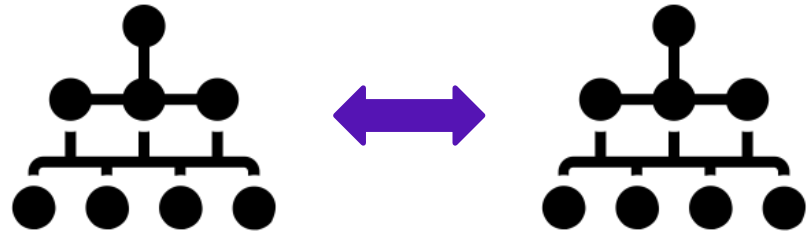
Initial Observations of AI-SE Integration Challenges – Detection & Action





Outline of a Research Agenda for MDE for SE & AI

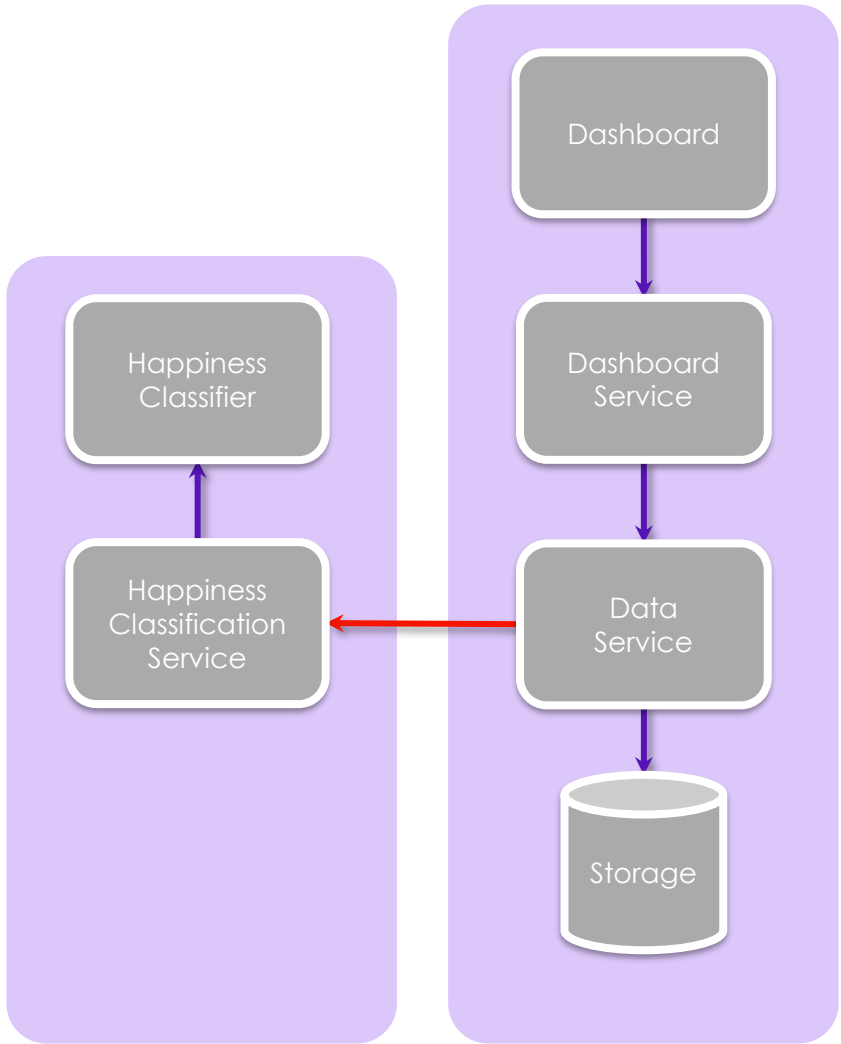
- AI and software developers will have to work together in the future
- Two conceptual domains that need to be aligned…
- … and extended based on the problem domain being addressed
- Potential challenge areas:
  - AI model observations and actions
  - Impact tracing across domains in response to observations and actions
  - Technical to AI challenges (e.g. security conformance, performance implications, etc.)
  - Release and documentation integration
  - Development process integration



© British Telecommunications plc 2020

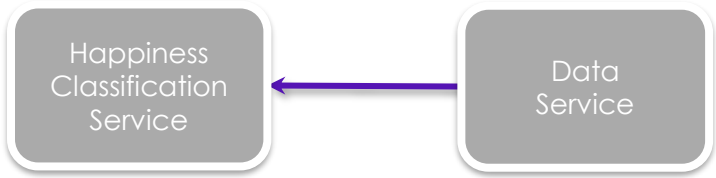
Not used / Supporting detail

Initial Observations of AI-SE Integration Challenges - Security



- The choice of AI delivery pipeline (e.g. Kubeflow) locks you in to they way they choose how to serve up the classifier
- Kubeflow currently does not have out of the box support for OAuth2
- The current system therefore has insecure communication
- Who should fix this problem?

Initial Observations of AI-SE Integration Challenges – Brittle Communication




```json
{
    "id": "1",
    "inputs": [
        {
            "name": "input",
            "shape": [
                1,
                2
            ],
            "datatype": "INT32",
            "data": [
                30,
                40
            ]
        }
    ],
    "outputs": [
        {
            "name": "predict"
        }
    ]
}
```

```json
{
    "waitTime": 14,
    "serviceTime": 20
}
```

- Data lifecycle pipelines take generic and brittle approaches to message exchange
- All requests are handled using untyped vectors rather than structured data
- Makes the overall system prone to (undetectable!) failures when evolving
- Who is responsible for fixing this?