Ontology-Driven Knowledge Graphs for Personnel Management Within the UK Ministry of Defence: A Conceptual Overview

Student: Jack Shufflebotham

Supervisor: Fanny Camelia, Tim Ferris

14-Nov-24

www.cranfield.ac.uk

Contents

Introduction - Context

- MODs has several initiatives to facilitate skill capture
- The MOD recognises the need for a “single version of the truth”
- Integrating skills data into other systems would enable better decision-making processes.
- Defence Systems Approach to Training (DSAT)

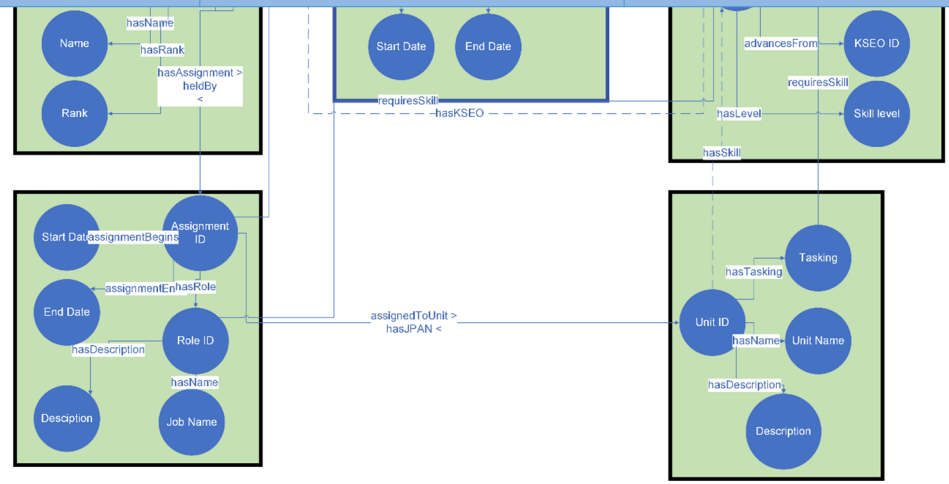
Introduction - Objectives

- Introduce the concept of Ontology-Driven Systems
- Review literature around the use of systems in other domains
- Propose a conceptual framework for an “Ontology-Driven Skill Management System”
- Defence Systems Approach to Training (DSAT)

Conceptual Model – Ontology Design and Reasoning

- Lightweight ontology – non-functional
- RDF – Subject, Predicate, Object
- OWL – Advanced reasoning
- Semantically linking data increases its utility
  - Big data analytics and machine learning
  - Explanation generation
  - Demonstration

Subject	Predicate	Object
:John_Doe	rdf:type	:MilitaryPersonnel
:John_Doe	:hasRank	:Sergeant
:John_Doe	:assignedToUnit	:101st_Airborne_Division
:John_Doe	:hasServiceNumber	"1234567"
:John_Doe	:stationedAtBase	:Fort_Campbell



Conceptual Model – System Architecture

- GraphDB
- Webserver – Python… Why Python?
- API endpoints
- Node based

```python
from flask import Flask, jsonify, request, render_template
from SPARQLWrapper import SPARQLWrapper, JSON
from geopy.geocoders import Nominatim

app = Flask(__name__)

GRAPHDB_ENDPOINT = 'http://LAPTOP-P2CJ8H9V:7200/repositories/DSTL'

geolocator = Nominatim(user_agent="Test_HR_App")

def get_lat_long(address):
    location = geolocator.geocode(address)
    if location:
        return (location.latitude, location.longitude)
    else:
        return

# Home Screen
@app.route('/')
def home():
    return render_template('index.html')


# SPARQL Query Interface
@app.route('/SPARQL', methods=['GET', 'POST'])
def SPARQL():
    if request.method == 'POST':
        query = request.form.get('query', '')

        if query:  # Check if the query is not empty
            sparql = SPARQLWrapper(GRAPHDB_ENDPOINT)
            sparql.setQuery(query)
            sparql.setReturnFormat(JSON)


            try:
                results = sparql.query().convert()
                return render_template('SPARQL.html', results=results)
            except Exception as e:
                # Pass the error to the template, if needed
                error = f"An error occurred: {str(e)}"
                return render_template('SPARQL.html', error=error)
        else:
            # Handle the case where the query is empty
            error = "Please provide a SPARQL query."
            return render_template('SPARQL.html', error=error)
```

Use Cases

- Skill management
- Recruitment
- Training management
- Career management
- Contingency planning (Strategic workforce planning)
- Capability management

Conclusion and Next Steps

Benefits

- Standardisation and integration
- Enhanced decision making
- Increased operational efficiency

Challenges

- Difficulties integrating with current MOD systems
- Data complexity and consistency
- Ethical and Privacy concerns

Future Direction

- Develop and validate a skill ontology against Defence needs as the backbone of the system
- Create a proof-of-concept demonstrator for the system
- Address challenges by integrating features such as “Explanation Generatoin” to remove the black box feel of AI

Any questions?