INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA DE INTELIGÊNCIA ARTIFICIAL




Melhorar a Gestão de Recursos Humanos através da Avaliação dos Conhecimentos e Competências dos Colaboradores

JOÃO SILVA MOREIRA
Outubro de 2022

P.PORTO

Improving Human Resource Management by Evaluating Employees' Knowledge and Skills

João Silva Moreira

Student number: 1160928

Dissertation to obtain the master's degree in

Artificial Intelligence Engineering

Academic Supervisor: Dr. Constantino Martins

Academic Co-Supervisor: Dr Luiz Faria

Jury:

President:

Dr. Isabel Cecília Correia da Silva Praça Gomes Pereira, Professora Coordenadora do Instituto Superior de Engenharia do Instituto Politécnico do Porto

Members:

Dr. Maria Goreti Carvalho Marreiros, Professora Coordenadora com Agregação do Instituto Superior de Engenharia do Instituto Politécnico do Porto

Dr. António Constantino Lopes Martins, Professor Adjunto do Instituto Superior de Engenharia do Instituto Politécnico do Porto

Porto, October 2022

Abstract

In today's world, organisations recognise the importance of human capital as their most valuable resource and are rewarded for the correct management of their human resources. This document showcases a system whose main goal is to better manage the development of collaborator's skills and optimise the allocation of human resources to projects within the organisation.

The state of the art was researched in the fields of user modelling, knowledge engines, classification algorithms, taxonomies of educational learning, and methods of measuring similarity. Systems that tackled some similar aspects to the system to be showcased were also presented. It was concluded that an overlay user model could be used to portray the knowledge of the collaborators using the system. Collaborator knowledge evaluations based on an dynamic questionnaires algorithm could be implemented using a knowledge engine. The system could calculate the engineering profile the collaborator was closest to attaining by using the Euclidean distance between vectorial representations of both.

The system was implemented as a component to be incorporated in a larger system. The larger system could provide ours with functionalities such as user authentication and collaborators' self-evaluation of their knowledge. The larger system in mind was the Capgemini Engineering's internal system, but it could be integrated with any other.

The user model is created based on the latest evaluation the collaborator made, including their own self-evaluation, if available. The dynamic questionnaires algorithm was implemented using Pyke, a Python knowledge engine library. The sci-kit learn library was used to implement the Euclidean distance.

The testing of the system was done in a two-round experimentation process where Capgemini collaborators attempted to complete a dynamic questionnaire evaluation. The system correctly inferred the knowledge of the collaborators thirty-eight percent of the time. As such, it was concluded that the evaluations are not trustworthy and must undergo some improvements and more experimentation.

Suggestions for future work on the system were made, mainly having the option to import questions from outside sources, possible experiments to help assess the system, and have the system suggest study materials according to the knowledge the collaborator lacks.

Keywords: collaborator, user modelling, knowledge engine

Resumo

Nos dias de hoje organizações reconhecem a importância do capital humano como o seu recurso mais valioso e são compensadas pela gestão correta dos seus recursos humanos. Este documento visa apresentar um sistema cujo objetivo principal é melhorar a gestão do desenvolvimento do conhecimento dos colaboradores e a otimização da alocação de recursos humanos a projetos dentro da organização.

Foi efetuada uma pesquisa sobre o estado da arte nas áreas de modelação do utilizador, motores baseados em conhecimento, algoritmos de classificação, taxonomias de objetivos educacionais e métodos de medir a semelhança entre representações de conceitos. Também foram apresentados alguns sistemas que abordavam aspetos semelhantes ao sistema a apresentar. Concluiu-se que um modelo de utilizador overlay poderia ser usado para representar o conhecimento dos colaboradores que usariam o ssitema. Avaliações do conhecimento de um colaborador baseadas num algoritmo de questionários dinâmicos poderiam ser implementadas usando um motor baseado em conhecimento. O sistema poderia calcular o perfil de engenharia cujo certo colaborador estaria mais perto de alcançar usando a distância Euclidiana entre representações vetoriais de ambos.

O sistema foi implementado como um componente a ser integrado num sistema ainda maior. O sistema maior poderia contribuir para o nosso com funcionalidades como a autenticação do utilizador, e auto-avaliações do conhecimento dos colaboradores. O sistema maior em mente seria o sistema interno da Capgemini Engineering, embora possa ser integrado com outro qualquer.

A criação do modelo do utilizador baseia-se na última avaliação reallizada pelo colaborador, incluindo uma auto-avaliação, se disponível. O algoritmo de questionários dinâmicos foi implementado usando uma biblioteca de motores baseados em conhecimento em Python, chamada Pyke. A biblioteca sci-kit learn foi usada para implementar a distância Euclidiana.

O sistema foi testado através de duas rondas de experimentação, em que colaboradores da Capgemini tentaram completar uma avaliação do seu conhecimento baseada em questionários dinâmicos. O sistema inferiu corretamente o conhecimento dos colaboradores trinta e oito por cento das vezes. Como tal, foi concluído que as avaliações não são confiáveis, e será necessário um melhoramento do sistema, e mais experimentação.

Foram feitas sugestões de trabalho futuro, principalmente a existência de uma opção para importar questões de fontes exteriores ao sistema, possíveis maneiras de testar o sistema, e uma funcionalidade que sugerisse material de estudo ao colaborador, conforme o conhecimento que lhe falta.

Palavras-chave: colaborador, modelação do utilizador, motores baseados em conhecimento

Acknowledgments

I greatly appreciate the help and advice provided by professors Constantino Martins, Luiz Faria, Marílio Cardoso and Paulo Matos throughout the whole duration of my participation in the HORUS project and the writing of this document. Their constructive criticism and explanations of their thought process were very insightful and helped me constantly improve my work.

A special thanks to my friends Fernando Alves, Ricardo Gil and João Leite for always being available to help discuss ideas and approaches regarding a certain problem, to listen during the times I was most frustrated, and to provide companionship during every step of the way.

Lastly, a thank you to everyone who cared enough to inquire about how this work was proceeding and my well-being.

Index

Figure Index

Table Index

Code Snippet Index

Acronyms and Symbols

List of Acronyms

AI - Artificial intelligence

1 Introduction

This chapter will focus on introducing the reader to the subjects this thesis will tackle. Starting with a description of the problem it tries to solve, segueing into the specific goals it tries to accomplish, the research questions posed, the contributions it brings to certain fields of study, the methodology used in the research, and finalising with the planning.

1.1 Problem Description

In today's world, companies recognise human capital as the most valuable resource for their organisation. Having employees that are qualified and motivated is a key component in being successful in a competitive market. This is especially true for businesses that are based on technology, for there is a need to keep up with something that is constantly changing and evolving [1].

Coinciding with these facts, companies are rewarded for managing their human resources correctly, keeping them one step ahead of the curve. This can be accomplished by continuously educating the employees, providing them with new skills and adapting them to an ever-evolving environment. Another approach is by making sure employees are optimally allocated to the positions the company needs filled, minimising the waste of human resources [2]. These approaches are not mutually-exclusive and can be applied at the same time.

The HORUS Project (Refª: POCI-01-0247-FEDER-72235) is a project [3] whose main objectives are: the investigation and development of a new platform that allows companies to carry out a real and efficient management over the development of collaborator's skills; the optimization in the allocation of human resources to their teams and projects; and the increase of effectiveness of selection processes. This solution should prove an essential tool to human resource managers in tech companies. The platform should have:

- A structured database containing the skills to be evaluated, and questionnaires and questions used to evaluate the employee's knowledge;

- An evaluation process, based on questionnaires, that can objectively evaluate employee's knowledge about a specific skill;
- An intelligent system that analyses performance based on the answers the employees give during a questionnaire, as they are given, allowing the questionnaire to adapt to the user;
- A system that recommends engineering profiles to be assigned to certain employees and identifies the employees that are closer to achieving the knowledge required to attain a certain profile;
- A crowdsourcing component that enhances the collection of content related to engineering profiles, skills, questions and study materials, in order to increase the robustness of the evaluation models and to allow the platform to evolve to support a greater number of engineering profiles and skills.

The HORUS Project serves as the main context in which the work that will be presented in this document was developed. The focus will be on the components responsible for evaluating an employee's knowledge regarding a certain skill, and the ability to assign engineering profiles to employees with the appropriate knowledge levels in the required skills.

1.2 Objectives

Considering the goals of the HORUS Project, the work presented will focus on: creating a system that evaluates collaborators' knowledge using dynamic questionnaires; and identifying which engineering profile is closest to their own knowledge. As such, we can define the main objective of this work is improving human resource management by providing an accurate evaluation of the knowledge possessed by collaborators so they can be more effectively assigned to projects within a corporation. This can then be divided into the subset of objectives:

- Modelling both the employee's knowledge and the knowledge necessary to obtain an engineering profile in a way that allows a comparison to be made;
- Identifying which engineering profiles are close to being attained by an employee, as to provide them a goal to work towards;
- Creating a system based on dynamic questionnaires that allow for an accurate evaluation of an employee's knowledge.

1.3 Research Questions

To accomplish the objectives defined in 1.2, the research questions that we will look to answer are the following:

- Can we model a collaborator's knowledge and the knowledge required to attain an engineering profile in a comparable form?
- Can we identify an engineering profile a collaborator is close to attaining?
- Can we accurately evaluate an employee's knowledge through the use of dynamic questionnaires?

These questions were identified as being able to help in determining if the system is working properly and fulfils its purpose. Each research question directly corresponds to one of the objectives detailed previously.

1.4 Contributions

This work will hope to contribute to the area of Artificial Intelligence by providing a solution that incorporates AI techniques applied in the context of human resource management in organisations. It also hopes to contribute to human resource management by helping with the continuous education of an organisation's collaborators, and the optimisation of the allocation of human resources to projects within said organisation. The system uses an overlay user model to depict the knowledge of collaborators. A knowledge engine is used to evaluate the collaborators' knowledge through a dynamic questionnaire algorithm. This is done in the form of the HORUS platform that will be able to:

- Evaluate employees' knowledge and skills;
- Adapt the evaluation questionnaire according to the employees' answers to ensure a better evaluation of their knowledge and skills;
- Assign engineering profiles to the employees so human resources can be more effectively allocated to new projects.

1.5 Planning

This project was divided into five different stages, as depicted in Table 1:

Table 1 - Stages of the project

Stage	Duration
Understanding the project	One month
State of the art research	Two moths
Design and Implementation	Seven moths
Experimentation	Two months
Document Writing	Twelve months

The first month after the author joined the HORUS project was dedicated to understanding the project, its goals, requirements, and intricacies.

During the second and third month research was conducted into the state of the art of the relevant and applicable techniques and technologies. The research was conducted in an informal fashion. Most of the research consisted in trying to find relevant articles that discussed the researched subjects, similar works where they were used and the appropriate tools to implement the techniques. The main sources used in this research were Google Scholar, B-On, ISI Web of Knowledge and the IEEE Xplore Digital Library. Throughout the research the keywords used were User Modelling, Student Modelling, Knowledge Modelling, Classification Algorithms, Knowledge Engines and Bloom taxonomy.

The design and implementation of the system was done consistently and non-stop during the next seven months. This included the design and implementation of the database, of the dynamic questionnaires algorithm and the system components. The design and implementation were done consistently and non-stop because as the system was designed and implemented there was constant feedback that then directly demanded changes in the design and implementation. As such, there was never an ending to both of these phases until the testing of the implementation began.

The last two months were dedicated to two rounds of experimentation of the system implemented and its assessment. Implementation ceased at the start of the first round of experimentation. After the end of this first round, considering the results of the assessment and the noted shortcomings of the system, slight changes were made to the implementation to get it ready for a second round of experimentation. The second round of experimentation resulted in an assessment that will influence what will be done in the future regarding this project.

The writing of this document took place throughout the entire research, development, and experimentation process.

1.6 Structure

The content of this document is organised as follows:

Chapter 2 shows the study and review of the state of the art on user modelling and student modelling, classification algorithms, knowledge engines, a brief overview of the bloom taxonomy and some implementations of similar systems.

Chapter 3 presents the design and implementation of the system and its components, the database used, the user modelling approach, the dynamic questionnaire algorithm created, and the tools used.

Chapter 4 contains the experimentations done to assess the system, how they were structured, what they tried to assess about the system, the tester's experience and the discussion of the results obtained.

Chapter 5 contains the conclusions of this project, the answers to the research questions, what was completed and the future work yet to be done.

2 State of the Art

In this chapter the state of the art will be discussed, in regard to various techniques that can prove relevant in the development of the system. Knowledge Engines, User and Student Modelling, Classification Algorithms, and the Taxonomies for educational learning. Knowledge Engines will be discussed as an hypothesised solution to the implementation of dynamic questionnaires. User and Student Modelling can be used as a way to model the knowledge of the collaborators. Classification Algorithms and techniques for measuring similarity are possibilities for calculating the engineering profile a collaborator is closes to attaining. Taxonomies of educational learning can be of use in modelling the domain knowledge. There will also be an overview of other work that tackles similar problems to the ones we are trying to solve, or some aspects of them.

2.1 Knowledge Engines

Knowledge engines are systems that try to mimic an expert's problem-solving skills in a certain area. By simulating the thought process of human experts, knowledge engines hope to output solutions comparable to what said experts would provide in the same scenarios. The two main components in these systems are the knowledge base and the inference engine [4].

Knowledge bases portray the separation of data from logic, and as such, are composed of facts and rules. The facts constitute the data that is known for a specific case, while the rules portray the thought processes that experts would use to draw conclusions or infer new facts about the case [5]. Facts and rules are usually obtained from a knowledge acquisition process with experts in the field the knowledge engine is being applied to. This knowledge acquisition process can be performed using methods such as eliciting knowledge from trusted sources or interviews with the experts [6].

Rule/Inference engines apply the rules depicted in the knowledge bases according to the facts. These engines must determine what information is needed, obtain that information, either by using rules to infer it or by asking the user directly, and use it so solve the problem it was tasked with providing a solution for [5].

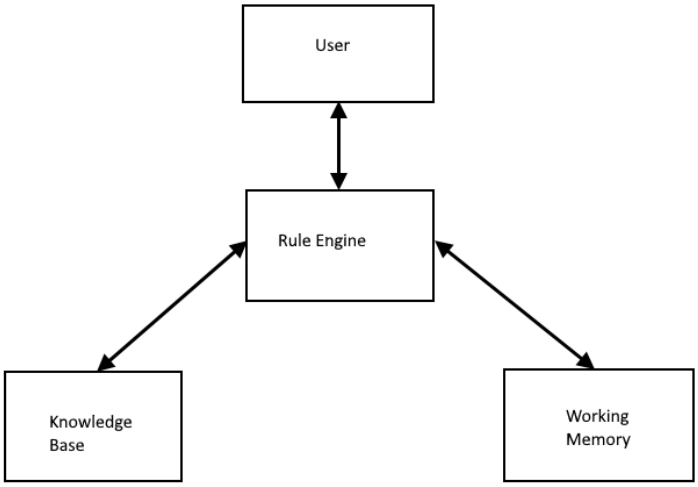

Figure 1 – General architecture of a knowledge engine based on [4]

Figure 1 depicts a general architecture of knowledge engines. The user interacts only with the Rule Engine component, through the use of a user interface. Engine which then uses the information stored in both the Knowledge Base (facts) and the Working Memory (data acquired through other means, such as asking the user directly or fetching information from the database), applies logic (rules from the Knowledge Base) to them and outputs a solution back to the user [5].

Other components that can be included in a knowledge engine are an explanation module, a user interface, and fuzzification and defuzzification modules for a knowledge engines based on fuzzy logic.

Explanation modules can provide to both users of the system and its developers, explanations as to how and why the system reached a conclusion. Meaning it can describe its reasoning, how a request related to a rule, how rules lead into one another and how and why rules are activated. The developers can then use the explanations to better understand its inner workings and to improve it. Users of the system can use this to understand why the system is asking a particular question to them or how did it reach the conclusion that it did [7].

A user interface is a part of any expert system that is based around interacting directly with the user. The communication between the user and the system is made bidirectional, and as the user must be able to clarify the problem they want solved, the system must be able to respond to their request [8].

Fuzzy logic works on the assumption that a proposition is only partially true, allowing reasoning based on partially true imprecise statements. As such, a knowledge engine that functions based on fuzzy logic has to have a fuzzification module that transforms the data the system is going to use into a fuzzy set ready to be inferred from, and a defuzzification module that in turn transforms a set of fuzzy conclusions that the system inferred into a single value [9].

2.1.1 Application of Knowledge Engines

Knowledge engines can be applied to multiple different areas. Some examples of the application of knowledge engines, focusing on the education and intelligent tutoring fields, follow.

In 2015, Cucu outlined how a knowledge engine could be implemented in an intelligent tutoring system in the domain of the education of communication skills [10]. In 2013, Grivokostopoulou, Perikos and Hatzilygeroudis presented an intelligent tutoring system that used a knowledge engine to teach students first order logic, a basic knowledge representation language. It was noted that future research could be focused on an automatic assessment mechanism that assessed the students during their interaction with the system [11]. In 2017, the same authors presented another knowledge engine capable of assisting in the learning and teaching of search algorithms by analysing students' answers in proximity to the correct answers [12]. In 2022, Bradáč et al. proposed a model of an Intelligent Tutoring System that would highlight the needs of the students so they could be addressed. The implementation used a fuzzy logic knowledge engine to assess students' knowledge in the studied area. From the output of the assessment the system knows what the student lacks and adapts the learning materials to meet their needs [13]. In 2020, Montalvo-Ochoa et al. presented an educational tool to aid in the training process related to water quality determination. The system implemented a knowledge engine designed based on the Water Quality Index international standard and Ecuadorian standard to determine water quality. Students can then use the functionalities of the system to learn how to conduct the studies on water quality, and test what happens when certain variables change [13]. In 2020, Lee et al. presented an expert system that would recommend university programs to students based on their characteristics. Each student's personality is examined according to Holland's Personality test. The system then updates the knowledge base with the results of the personality test and infers a recommendation for a university program accordingly [15].

2.1.2 Knowledge Engine Technologies

There exist multiple technologies dedicated to the implementation of knowledge engines in various systems. Researching in an informal fashion yielded a few different technologies to be discussed. Following in Table 2 is an overview of some of the existing ones found during research.

Table 2 - Knowledge Engine technologies comparison

Technology	Language	Availability	Algorithm	Other Features
Drools [16]	Java	Open-source	Rete	Provides its own Eclipse-based IDE
PyCLIPS [17]	Python	Open-source	Forwards-chaining rules, Rete	Implements CLIPS as a separated engine
SWI-Prolog [18]	Prolog	Open-source	Anything	Very flexible, allows for the implementation of anything
Pyke [19]	Python	Open-source	Forwards/Backwards-chaining rules	Asks user questions

Drools provides an inference engine capable of processing rules and facts to produce an output. It tries to bridge the gap between business and technical teams by having very easy to understand syntax to write the rules in. Rules are written in a drools file (".drl" extension) in the form of a "When-Then" construct, where the "when" section lists the conditions for the rule's activation, and the "then" section lists the actions to be taken upon activation. It also has the capability of defining rules in a pre-formatted Excel spreadsheet. In 2013, Jaques et al. used an expert system implemented with Drools as part of an Intelligent Tutoring System that corrected students' exercises about algebraic equations [20].

PyCLIPS is a Python module that allows to use a functioning CLIPS engine in a Python program. CLIPS is a pattern-matching rules language written in C that uses forward-chaining rules. It provides access to CLIPS' internal functions and structures through the use of its own classes and top-level functions. It also allows for the calling of Python code within the CLIPS subsystem. In 2012, Smith Jr. showcased an expert system, implemented using PyCLIPS, that aided in the cartographic design process by providing starting points for the mapping of an area [21].

Swi-Prolog offers an environment for programming in Prolog, a logic programming language. As such, being such a basic technology, it allows for the flexibility of implementing the knowledge engine however we desire. In 2013, Singla presented a medical expert system that would help in the diagnosis of lung diseases in patients. The system was implemented in Prolog and had a seventy-percent success rate in its diagnoses [22].

Pyke, inspired by Prolog, introduces a form of logic programming in Python. It allows for the definition of rules and facts in specific files in a syntax akin to logic programming, which are then converted into Python code by Pyke. A way to invoke Python code from the system's rules is also provided, thus yielding greater flexibility. Finally, it possesses a functionality that allows for the asking of questions to the user directly from rule activations. In 2018, Zhang et al. proposed an expert system for local flood forecast and warning. Implemented in Pyke, the system was capable of providing its reasoning, forecast on the flood warning stages, possible consequences and recommendations for the public [23].

2.2 User Modelling

It is assumed that Intelligent Tutoring Systems should adapt to learners and their individual characteristics. This can be achieved by changing the contents presented, changing the way the interaction with the user is performed, taking into consideration state of the learner, among others. These techniques can translate into both a better performing tutor and learner [24]. In our case this would translate to a variation in the questions that are presented to the users according to the answers given during an evaluation, so the system can better fulfil its purpose of correctly estimating the user's knowledge.

In order for the system to be able to adapt to specific learners, it must first be able to understand them. User modelling is a technique that tries to define and understand users, by gathering information about them and detecting patterns during their interactions with the system [25]. The data gathered can be a different combination of the user's interests, preferences, knowledge, background, goals, skills or individual traits, depending on what the purpose of the user model is and the intent behind its creation [26].

User models can be classified according to six different dimensions:

- Individual or group profiling: whether the user model is built for a specific individual or for a group of similar individuals, by the means of overlapping some information between the members of the group [27]
- Supervised or unsupervised learning: supervised profiles use data mining to recognise patterns in a labelled dataset, while unsupervised profiles try to identify new knowledge in unlabelled data [26]
- Explicit or implicit feedbacks in user models: explicit feedback will be given by the users explicitly and of their own volition, implicit feedback is collected by analysing the actions of the users [28]
- Long-term or short-term user models: while short-term models focus on the users current interests, long-term models consider the end-goals [29]
- Static or dynamic user models: static profiles never change once created, contrasting dynamic profiles that, because of the ever-changing nature of a user, adapt to users' new interests and goals [30]

- Distributive (decentralised) or non-distributive (centralised) user models: whether the user modelling component is embedded into the application or if it is a separate application [31]

2.2.1 Student Modelling

The users of our system are employees of a company trying to learn new skills, and as such we can say that, in this context, they are similar to students. Student modelling is the usage of user modelling in the context of applications of computer-based intelligent instructional systems, and as such, more appropriate to attaining the goals described in 1.2. "A student model is a representation of the system's beliefs about the learner and is, therefore, an abstract representation of the learner in the system." [32] Student modelling differentiates itself from User modelling by mainly focusing on representing the students in a manner related to the pedagogical objectives set, like the skills they need to learn and how much of those skills have they learned/mastered, while User modelling is more general and can depict many more aspects of the user [33].

Human teachers learn how to best teach their students based on their years of experience. They pay attention to the students' facial expressions and body language, learn what approaches work best for each student, repeat materials and correct misunderstandings to make the students better understand them. Teachers also interact with their students to try and get information on their motivations, goals, and interests. Intelligent tutoring systems try to emulate these behaviours and make inferences about the student's knowledge, preferred learning styles, among others, to build an individual student model for every user [34].

2.2.1.1 Student Modelling Approaches

Student Modelling can be approached in multiple different ways, varying in complexity and characteristics in accordance with the desired purpose for the student model. Approaches change in answer to the questions of what to model, how to model it, and why [35], [36]. Following are some examples of student modelling approaches:

- Overlay Model – Considers the student's knowledge to be incomplete but correct, therefore it is represented as being a subset of the domain knowledge. Meaning for each fragment of the domain knowledge, an overlay model would depict the subset of this knowledge the student already possessed. This subset can be represented as a set of true/false values per fragment, or as the degree as to which the student understands a fragment of the domain, through the use of qualitative indicators (good, average, poor,…) or quantitative ones (probability that a student understands the concept). The domain must be able to be represented in the form of different individual topics and concepts, allowing for its division into smaller fragments. As such, the complexity of the model is highly dependable on the granularity of the domain structure [36], [37].
- Perturbation Model – Is an extension of the overlay model that also takes into account the possible misconceptions, or incorrect knowledge, the student possesses. It considers the student's knowledge a perturbation of the expert's, allowing for the

depiction of both correct but incomplete knowledge, and incorrect knowledge and assumptions. This portrayal of incorrect beliefs allows the system to address them, and correct them, transforming them into correct knowledge about the subject. When evaluating a student, the system knows of the paths that might lead the student to give a wrong answer. By knowing which path the student took to give their wrong answer, the system will know how to tackle their incorrect reasoning. The wrong paths are usually collected in a bug library, built by an empirical analysis of mistakes (enumerative modelling), or by generating mistakes from a set of common misconceptions (generative modelling) [36], [37]

- Differential Model – Compares the behaviour of the student to the behaviour of an expert when faced with a similar problem and uses the differences to infer what the student needs to learn. Important domain concepts are considered "issues", which in turn determine what parts of the student's behaviour will be monitored. The model keeps track of the activated "issues" to determine what concepts is the student not applying correctly. The existence of an expert is required to compare the student's behaviour to [38]
- Constraint-based Model – Considers the domain's knowledge as a set of guidelines or constraints that must be followed to obtain the correct solution to a problem. Constraints are composed of a relevance and a satisfaction clause. The relevance clause dictates the condition that when met means the constraint is applicable. The satisfaction clause dictates the condition that must be satisfied for the constraint to not be violated. Tracking violated constraints will lead to the identification of the errors the student is committing [39]–[41]
- Stereotype Model – Clusters students into stereotypical groups, based on shared characteristics among them. Stereotypes have a set of trigger conditions that when met include a new student into the respective group. Systems will usually continuously check the student's responses, detect and resolve the conflicts between stereotypes and specific user knowledge values, and then update both the stereotypes and the student models. This model allows the inferring of a student's knowledge according to the stereotype that they are included in, thus avoiding the process of eliciting knowledge from every individual user. Finally it also lets the system adapt to new users at a much faster speed [36], [42], [43].

2.3 Classification Algorithms

In this work, we will try to assign engineering profiles to employees based on their skillset, which means we are dealing with a classification problem.

A classification problem is a problem that consists in identifying to which categories data belong to. These categories can be either pre-determined or discovered during the process of classification. Classifying data into categories is accomplished through the use of classification algorithms. These algorithms are used to assign a specific label to data based on its features [44].

2.3.1 Application of Classification Algorithms

Classification algorithms can be used to accomplish a variety of tasks in the education field, from predicting student performance, to identifying the factors that lead off-campus students to ride-source. Some examples of how classifying algorithms can be used in the education field follow.

In 2019, Cardona and Cudney used Support Vector Machines to predict degree completion in three years [45]. Also in 2019, Wiyono and Abidin compared the K Nearest Neighbours, Support Vector Machines and Decision Trees algorithms in predicting students' performance [46]. In 2018, Slim et al. used Logistic Regression and Support Vector Machines to predict student enrolment and identify the factors that had the highest correlation with an applicant's decision of enrolment [47]. In 2020, Aghaabbasi et al. used Random Forests and Bayesian Networks to identify the factors that lead to ride-sourcing, and the frequency of its usage [48]. In 2020, Hasan et al. proposed a supervised data classification model that tried to predict the academic performance of students at the end of a semester. The model was tested with several classification algorithms and concluded that the Random Forest algorithm performed the best [49]. In 2021, Hao and Chai experimented with combining the Support Vector Machines and K Nearest Neighbours algorithms to construct an intelligent classification model. This model, applied to the field of IT vocational education, would collect students' data for simulation analysis and predict their graduation status. It was concluded that this implementation had good results for application in the real world to help improve learning quality [50]. In 2018, Pérez et al. elaborated on a case study that focused on predicting dropouts of undergraduate students after six years of enrolment in a Colombian university. The experimentation used various algorithms, including Decision Trees, Logistic Regression, Naïve Bayes and Random Forests [51].

2.3.2 Classification Algorithm Techniques

There are multiple classification algorithms, varying in accuracy and performance. Some of this variance can be attributed to the quality of the algorithm itself, as some have become less used due to better alternatives showing up, but it can also be due to the data itself, as some algorithms work best with specific types of data [52]. Here are some examples:

Logistic Regression: Logistic Regression models the probability of a certain class given some data. It uses logistic functions to estimate these probabilities that in turn serve as a measurement of the relationship between a dependent variable and one or more independent variables [44], [53], [54].

Naive Bayes Classifiers: Naive Bayes Classifiers are based on Bayes' Theorem, which is a mathematical formula used for calculating conditional probabilities [55]. It works under the naive assumption of independence among the data's features when given the class label, which isn't always the case [44].

Support Vector Machines: Support Vector Machines perform classification by constructing an N-dimensional hyper plane, which then separates the data based on their target categories. A constructed hyperplane has a margin on both its sides, by maximising this margin the algorithm reduces an upper bound on the expected generalisation error [56], [57]. The model complexity of an Support Vector Machines is unaffected by the number of features encountered in the training data, and as such, easily deal with learning tasks that contain a large number of training features [58].

K Nearest Neighbours: K Nearest Neighbour classifiers label data in accordance with the most common class among a number ("K", which is usually a small positive integer) of the data nearest neighbours, the idea being that data that are very similar will belong to the same class [44], [59].

Decision Trees: Decision Trees are trees that label the data by sorting them based on their features. Data starts at the root node of a tree and sorted based on a value of a certain feature. This process continues for every subsequent node in the tree until a leaf node is reached, which then outputs the resulting label [44], [58], [60].

Random Forests: Random forests are a composed of an ensemble of decision tree predictors [61]. The decision trees are constructed at training time, based on the values of a random vector sampled independently, and with the same distribution for all trees in the forest [44], [61]. The output is the mode of the individual trees' outputs. Random Forests also account for the decision trees' tendencies to overfit their training dataset by performing small corrections [44].

In 2017, Zhang et al., compared the "classification prediction performance and time efficiency of eleven state-of-the-art classification algorithms, using publicly available data sets from UCI, KEEL, and LibSVM repositories." These eleven algorithms were Support Vector Machines, Random Forests, C4.5, AdaBoost, K-Nearest Neighbours, Logistic Regression, Stochastic

Gradient Boosting Trees, Extreme Learning Machine, Sparse Representation based Classification, Deep Learning and Naive Bayes Classifier. It was concluded that Support Vector Machines, Random Forests and Stochastic Gradient Boosting Trees were the top performers [44].

Also in 2017, Akinsola et al., compared seven classification algorithms, including Decision Trees, Random Forests, Naive Bayes, Support Vector Machines, Neural Networks (Perceptron), JRip and Decision Tree (J48). The research data was obtained from the National Institute of Diabetes and Digestive and Kidney Diseases. The research concluded that Support Vector Machines, Naive Bayes and Random Forests provided the highest precision and accuracy regardless of the number of attributes and data instances [60].

Works from Lessmann et al.(2015) [62] and Fernández-Delgado et al. (2014) [63] also compared multiple different classification algorithms, using a variety of datasets, with both Support Vector Machines and Random Forests ranking as the best predictors. Multiple other studies shown in [44] also consistently show these two as part of the top.

2.4 Measuring Similarity

Classification algorithms usually require a dataset with to be effective [64]. Considering we don't have access to one, there must be an alternative approach to calculating the engineering profile a collaborator is closest to attaining. As such, various possible approaches to measuring similarity between an engineering profile and a collaborator will be discussed.

2.4.1 Euclidean Distance

The Euclidean distance uses the Pythagorean Theorem to calculate true straight line distances between points in an Euclidean space (a finite n-dimensional real vector space intended to represent physical space) based on their Cartesian coordinates, according to the following equation [65]:

$$Euclidean\ Distance\ = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

In 2022, Supanich and Kulkarineetham presented an intelligent tourism recommender system that recommended tourist attractions to its users. The system took into account the user's sociodemographic and travel behaviour, calculated the most similar users, and recommended attractions that it deemed of shared interest. The Euclidean distance, cosine similarity and Pearson correlation measures of similarity were tested, with the conclusion that the Euclidean distance's results had the highest accuracy [66].

Also in 2022, Kumar et al. presented a medical consultation recommendation system meant to provide users with a self-diagnosis service. The user inputs the information such as their own

physical characteristics, clinical history and their symptoms. The system then compares this information to medical reports that concluded that a patient had a specific disease and calculated the three most similar to the data provided by the user using the Euclidean distance. Lastly, the system provided the user with three possible diagnosis based on the conclusions of the three most similar medical reports [67].

2.4.2 Cosine Similarity

Cosine similarity uses the angle between two vectors to calculate the similarity between them. Two vectors with an angle of zero degrees between them will have a cosine value of one, while any other amplitude different from zero will yield a value of less than one, therefore we can conclude that the lower the value the less similar two vectors are. Since it does not take into account the weight of the vectors when calculating similarity, it is only used when this aspect does not matter [68].

In 2021, Hartanto et al. presented a system that would detect plagiarism in thesis documents. The system would analyse the documents using Natural Language Processing and create term frequency vectors of the n-grams used in them. The similarity was measured by comparing these term frequency vectors using cosine similarity [69].

In 2022, Januzaj and Luma presented a system that calculated the similarity between higher education programs and job market demands. The system would build vectors with values corresponding to how many times certain keywords appeared in the documents it checked, then compared those vectors using cosine similarity [70].

2.4.3 Pearson Correlation Coefficient

The Pearson correlation coefficient is a measure of linear correlation between pairs of continuous variables. The higher the correlation coefficient the stronger is the relationship between data. The correlation coefficient is calculated using the following equation, where "cov(_)" refers to the covariance of the sample, and "var(_)" refers to the sample variance [71]:

$$Correlation\ coefficient = \frac{cov(x, y)}{\sqrt{var(x)} \times \sqrt{var\ (y)}}$$

In 2021, Devika et al. proposed a system that would recommend books to people in order to help them find books that would interest them most and keep their interest in reading high. The system would predict what a user would like to read based on the ratings given to certain books by users with similar interests. The system would determine these users with similar interests by comparing the ratings each user gave to the books read using Pearson correlation and cosine similarity [72].

In 2021, Sadia et al. presented a system that would recommend the appropriate fruit farmers should cultivate in specific soils. The system would take into account the soil's type and

geological formation and calculated the similarity between it and the fruits' cultivation requirements using Pearson correlation [73].

2.5 Taxonomies of Educational Learning

In order to portray the knowledge held by a collaborator, one must choose a suitable scale to adopt for this purpose. Taxonomies of educational learning allow for a depiction of a collaborator's knowledge in a uniform scale that can then be quantified and compared [74], [75].

2.5.1 Bloom Taxonomy

The Bloom taxonomy is a set of three taxonomies of educational learning objectives in different domains: the cognitive, affective, and psychomotor. The main purposes for the Bloom taxonomy is to help teachers discuss their problems with greater precision, to ease the discussion on curricular developments, among others [74], [76].

The affective domain of the taxonomy relates to the individual's relationship to the learning process, including aspects such as their interest, attitudes and reception of the subjects being taught [74], while the psychomotor domain relates to the individual's development of the skills required to perform a certain task, including aspects such as the skilful handling of a tool or instrument or the development of motor abilities [76].

The cognitive or knowledge-based domain of the taxonomy is the most relevant domain. It relates to the understanding of knowledge and the development of an individual's intellectual abilities [74]. It was originally divided into six levels of knowledge-based goals that represent the various states of understanding someone can have about a subject. The first level was Knowledge, which mainly involved the individual's ability to remember the specifics of a subject, the terminology used, conventions, the methods and processes involved. The second level, Comprehension, refers to the individual's ability to understand what is being communicated to them, translate those ideas into common language, and summarising concepts by stating the main ideas behind them. Application, the third level, involves the individual being able to apply the abstract concepts they have learned to concrete situations they encounter. The fourth level, Analysis, refers to the individual's ability to deconstruct complex ideas into simpler parts, as well as understanding the roles of each part and the relationships between them. The fifth level, Synthesis, involves the ability for the individual to form a plan or structure by combining and arranging various different parts or elements. The sixth and final level, Evaluation, takes into account the individual's ability to judge the quantitative and qualitative value about the extent of which ideas and methods fulfil certain criteria [74]. It was later revised in [77] to look like the following shown in Figure 2.

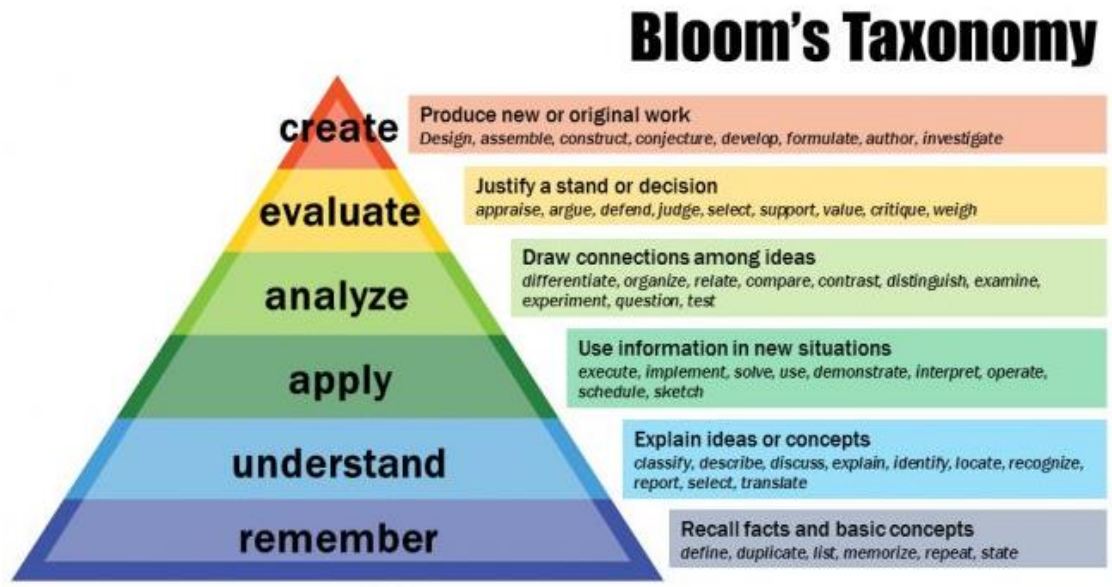

Figure 2 - Revised Bloom's taxonomy [78]

2.5.2 Fink's Taxonomy

Fink's taxonomy was created as an intended successor to the Bloom's taxonomy and designed to be able to help teachers decide what they wanted students to learn from their course. Dubbed "taxonomy of significant learning", it took into account the need for types of learning that did not come easy to the Bloom's taxonomy. "For example: learning how to learn, leadership and interpersonal skills, ethics, communication skills, character, the ability to adapt to change, etc." [75]

Unlike Bloom's taxonomy, it is non-hierarchical, and each of its aspects interact and feed of one another to stimulate other kinds of learning. Since its main purpose is to help teachers design their courses it is divided into six aspects, seen in Figure 3, all of which detail what the student should learn [75], [79]:

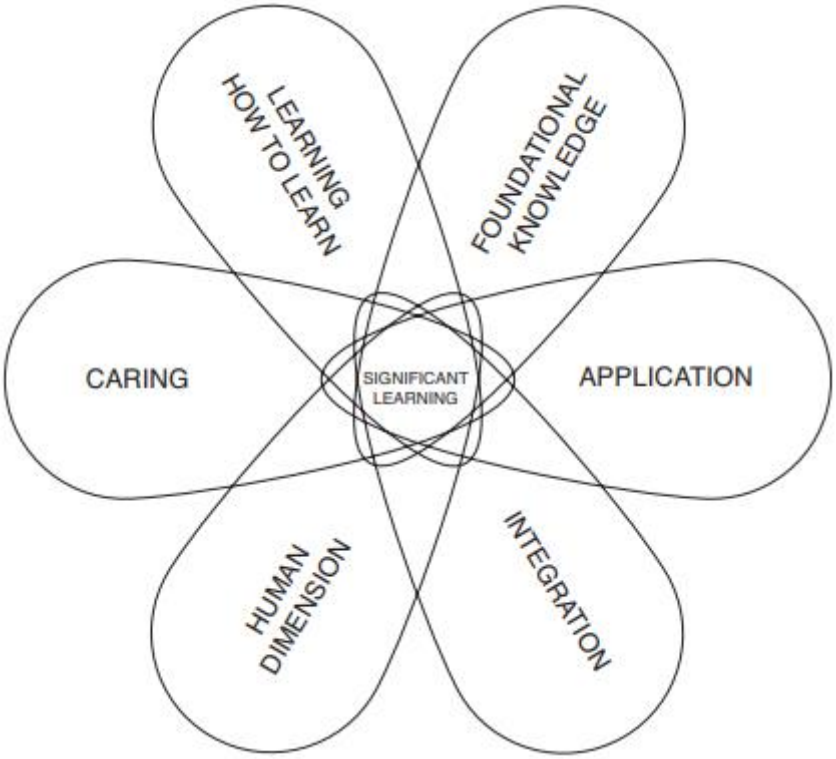

Figure 3 - Fink's taxonomy of significant learning [79]

- Foundational Knowledge – What concepts, ideas, facts, terms, relationship and other content that we want the student to understand and remember when the course is over [75], [79].
- Application – What it is that we want the students to be able to do by the end of the course, the skills that are important for the student to learn, the kinds of thinking they should be able to do (critical, creative and/or practical), and what complex projects they should learn to manage [75], [79].
- Integration – What connections, relationships and interactions should the student recognise in the real world, either between information, lived experiences, ideas or people [75], [79].
- Human Dimension – What should the students learn about themselves or about interacting with others they might encounter in the future [75], [79].
- Caring – "What new feelings, interests or values should the student develop from the course"[75], [79].
- Learning how to learn – What should students learn that will help them learn by themselves in the future, how to become a better student, and engage in discussions to acquire more knowledge in a subject [75], [79].

2.6 Knowledge Evaluation System Implementations

In 2014 P. Molins-Ruano et al., from the Universidad Autonoma de Madrid, proposed a computer assisted model for the evaluation of programming skills in students. The model was composed of both multiple-choice and completion type questions. Each question would correspond to a knowledge level so the difference between basic and advanced knowledge would be clearer and the assessment more effective. Every student would start with a basic level question and answer the same number of questions. When a student would consistently give correct answers to basic questions, more advanced questions would be asked. However, if the student failed to answer a question of a certain level, the next question would belong to the same or lower level of knowledge. The authors concluded their study affirming that the model would give objective, secure and complete assessments [80]. This model would end up being used in the e-valUAM application, a free software based on adaptive tests that has the goal of improving the quality of the tests in the Universidad Autonoma de Madrid by means of increasing the objectivity, solidity, security and relevancy of its contents [81]. In 2016, the authors expanded their work, this time trying to assess the students' knowledge by comparing their responses to answers given by experts in the field of Computer Science [82].

PDinamet is a web-based adaptive learning system for the teaching of physics in secondary education, which uses an overlay model in order to provide effective and personalized selection of the appropriate learning resources. During the year teachers had access to this system, the number of students who did not pass the course had notably decreased [36], [83].

An adaptive tutoring system that uses stereotypes in order to provide an individualized learning environment is CLT, which is a C++ tutor. The system used cognitive aspects of the user to present a completely individualised teaching environment for each student. By testing the system on freshmen students, the authors concluded that: students who were given contents based on their cognitive abilities performed 12% better than others, students with high cognitive abilities spent one to two hours less completing assigned tasks when presented with lecture contents according to their abilities, and every student who used the system outperformed students who were taught in the classroom [36], [84].

In 2013, Jaques et al. presented an Intelligent Tutoring System that corrected students' exercises regarding linear and quadratic equations. It used an expert system that compared the students' answers to the desired ones and using the differences to update the student model with the skills the system inferred the student needed to work on. The implementation could also demonstrate to the student how to solve a problem. The system was experimented on forty-three students, twenty-two of these had access to and used the system, and the other twenty-one did not. The experiment concluded that one could say with ninety-percent confidence that students who used the system scored higher than students who did not. Fifty-two percent of the students who used the system stated that with its help, their knowledge of algebraic equations increased [20].

In 2010, Baschera and Gross used a perturbation student model for spelling training, which represented students' strength and weaknesses, in order to allow for appropriate remediation actions to adapt to students' needs. Two user studies conducted with this system revealed that

students could progress at more than double the rate of normal students, mainly induced by the student adapted remediation actions [36], [85].

Adapquest, is a software tool for the development of dynamic questionnaires based on Bayesian networks. The system considers a questionnaire to be a Bayesian network, where each node in the network has multiple questions that can be asked. The questions are chosen by the Bayesian network model based on the previous answers the user gave. This tool was tested in a case study where it created dynamic questionnaires to help in early detection of employees at risk of mental health problems [86].

Table 3 presents a comparison of the showcased implementations. Doing a quick analysis one can conclude that, from the research done, systems that combine user models, expert systems and dynamic questionnaires in the education field do not exist.

Table 3 - Comparison of presented systems

System	User Model	Expert System	Dynamic Questionnaires
P. Molins-Ruano et al.	No	No	Yes
PiDinamet	Overlay Model	No	No
CLT	Stereotypes	No	No
Jaques et al.	No	Yes	No
Baschera and Gross	Perturbation Model	No	No
Adapquest	No	No	Yes

2.7 Summary

This chapter presented an overview of techniques used in the intelligent tutoring systems field. Examples of some applications of User Modelling, Classification Algorithms and Knowledge Engines were provided, as well as a brief overview of some Taxonomies of educational learning, and multiple implementations of similar systems.

The best knowledge engine technology for the task at hand seems to be Pyke, as it is a python library and has a functionality to ask the user questions, which goes hand-in-hand with the goals of the HORUS project to evaluate the user's knowledge using dynamic questionnaires.

From the student modelling approaches expounded, one can conclude that the overlay model is the one best suited for the system to be implemented. The concept of fractions of domain knowledge is a perfect fit with the knowledge levels of skills in the HORUS project. The domain of the HORUS project is also very divisible into skills, topics and subtopics, furthering the case for the overlay model is the best decision. If in the future the system was to expand into

correcting user errors the model could be easily expanded into a perturbation model, as this one is already an extension of the overlay model.

From the comparisons of the various existing classification algorithms put forward previously, one can conclude that the Support Vector Machine and Random Forests techniques were consistently placed as the top-performers. As such, these should be the methods to be tested if a classification algorithm was to be considered to be implemented. From these two, Support Vector Machines is the most indicated for smaller datasets, and Random Forests for large ones.

In the absence of datasets, the best approach to calculate the engineering profile a collaborator is closest to attaining seems to be by measuring the similarity between the collaborator's current knowledge and the knowledge required to attain an engineering profile using the Euclidean distance between them in an n-dimensional space. Cosine similarity is not a good fit for the problem as it does not consider the weight and magnitude of the vectors its comparing, only the angle between them.

From the educational learning taxonomies expounded, the Bloom taxonomy seems to be the most adept for the HORUS project, as it is better at classifying the student's current knowledge in specific skills, while the Fink's taxonomy is better at course creation and design.

Lastly, from the research done, we can conclude that there are no similar solutions to the one proposed. None that incorporate user modelling, knowledge engines and dynamic questionnaires in a single system, thus making the proposed system an innovation.

3 Design and Implementation

This chapter will present the how the system was designed and implemented, as well as the techniques and tools used. We will start by presenting the domain model and core concepts used. Then, we will detail the functional requirements and use cases. Following will be an overview of the system and its components. Lastly, we will proceed to explain every component in more detail, focusing on their purpose and implementation.

3.1 Domain Model

The Domain Model represents the core concepts used by the system, and the relationships between them. This Domain Model in Figure 4 is part of a larger model used for the HORUS project, containing only the concepts and relationships relevant to this system.

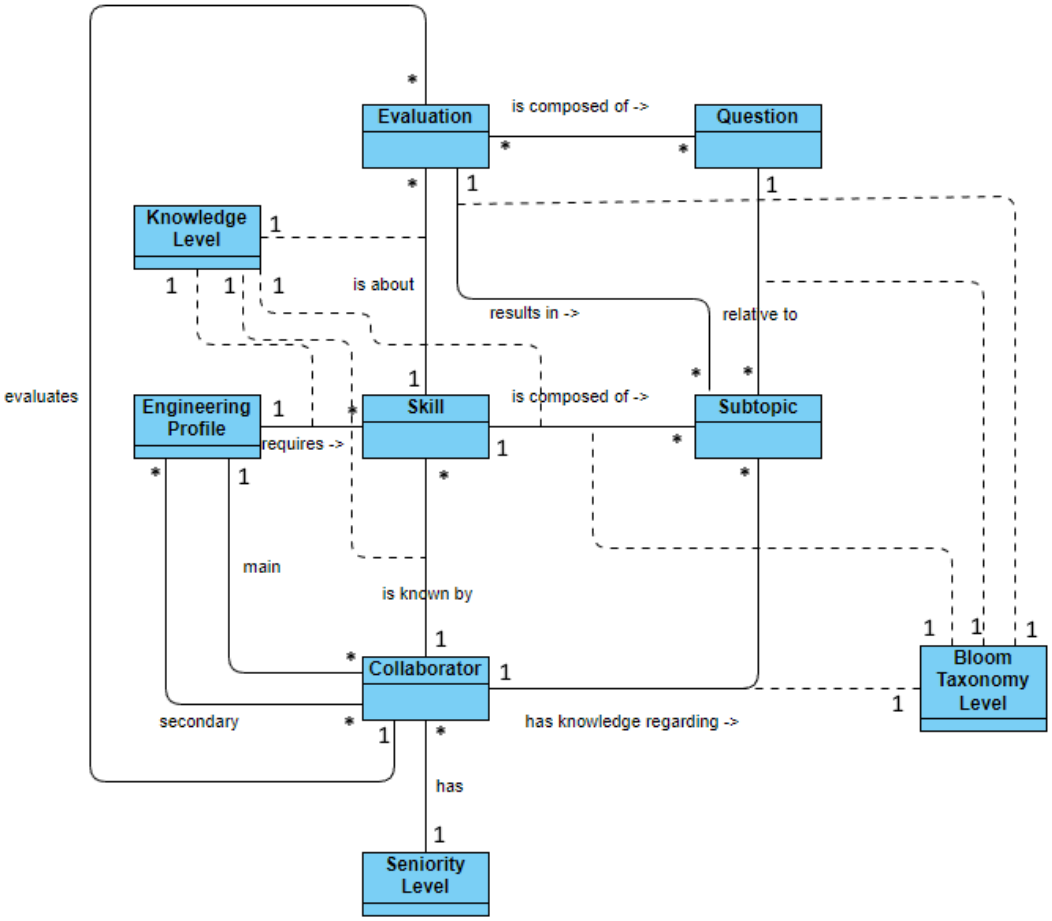

Figure 4 - Part of HORUS' Domain Model

The Collaborator is a user of the system, they can access the functionalities of the system, and a user model is built around their knowledge. A collaborator can hold knowledge about multiple skills, each classified with a knowledge level. Similarly, they can hold knowledge regarding several subtopics, each classified with a Bloom taxonomy level. Collaborators are classified within the organisation as having a seniority level. They can also have several engineering profiles, a main one and multiple secondary. Collaborators can perform several evaluations.

The Seniority Level is a classification of the collaborator within Capgemini Engineering. It can take the values of “Jr Cons./Tech”, “Cons./Adv. Tech”, “Adv. Cons.”, “Sr. Cons.” or “Expert”. Multiple collaborators can have the same Seniority Level

An Engineering Profile is a set of skills collaborators can have that influence how they are assigned to projects within the organisation. The skills required to attain an engineering profile are classified with a required knowledge level. Engineering profiles can be assigned to multiple collaborators as either their main or secondary profiles.

The Skill is a broad competency that can be held by any collaborator and be required to attain any engineering profile. A skill, matched with a specific knowledge level, can also be the chosen subject of several evaluations. Skills are composed of subtopics.

The Knowledge Level is a scale composed of four levels: “Beginner”, “Intermediate”, “Advanced” and “Expert”. These are used to evaluate knowledge regarding a certain skill.

A Subtopic is a subset of a skill, essentially a division of the broader competency into more specific competencies. The same subtopic can be a constituent part of multiple skills. Each of the subtopics corresponds to a specific knowledge level in a skill and is evaluated into one of the first five levels of the Bloom taxonomy. Subtopics can be part of multiple evaluations, and they can be the subject of several questions.

The Bloom taxonomy, described in 2.5.1, serves as a metric to evaluate knowledge regarding a subtopic.

A Question is used to evaluate knowledge regarding certain subtopics. It can be part of multiple evaluations and refers to one or more subtopics, each classified with a level of the Bloom taxonomy.

An Evaluation is a dynamic questionnaire done by a collaborator, regarding a certain skill, for a specific knowledge level. Evaluations are composed of multiple questions, and result in Bloom taxonomy levels for each subtopic of the evaluated skill.

3.2 Functional Requirements

The system must have a number of functional requirements, detailed as follows:

- R01 - It must be able to assess the collaborators’ knowledge in a certain skill
- R02 - The evaluation of the collaborators’ knowledge must be done by using a dynamic questionnaire
- R-03 The system must be able to calculate the engineering profile a collaborator is closest to attaining

Keeping these functional requirements in mind, Figure 5 depicts the use cases a collaborator can perform while using the system.

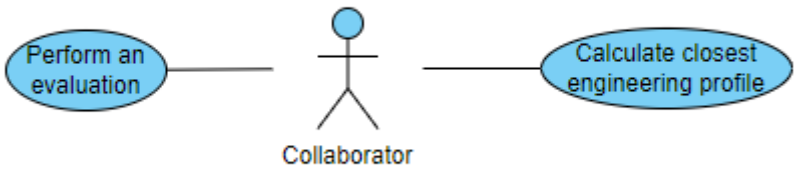

Figure 5 - Collaborator's Use Cases

UC01 - Perform an evaluation: Collaborators must be able to perform an evaluation of their knowledge in a specific skill. Evaluations are performed using dynamic questionnaires. The results of the evaluation will then be used to update the user model.

UC02 - Calculate closest engineering profile: Collaborators must be able to calculate which engineering profile they are closest to attaining.

3.3 System Overview

The system showcased in this document is but a component to be used by Capgemini Engineering's Internal System, according to Figure 6. This component Capgemini's system already has functionalities such as user authentication and user self-assessments which can then be used by this system to allow users to access its functionalities or to get the results of a collaborator's self-assessment.

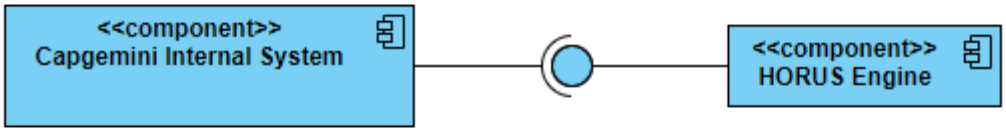

Figure 6 - Interaction with Capgemini's Internal System

The HORUS Engine is composed of seven main components, one of each is composed of three others, as depicted in Figure 7:

- the Database, responsible for storing all the information necessary for the system to perform any of its functionalities;
- the Database Access performs any operation that accesses the database. It also abstracts the HORUS Engine and the User Modeller from the Database implementation,

so if the implementation changes, no changes need to be made to both components that need it;

- the HORUS Engine is the core component of the system, it is the one that controls the flow of the program and links all other components together. It also provides a gateway for other systems to be able to use this system;
- the User Modeller is able to create a user model of a collaborator, and make any necessary changes to it;
- the Profile Distance Calculator determines the engineering profile a collaborator is closest to attaining using the sci-kit learn library;
- the Pyke Knowledge Engine is the component that runs the dynamic questionnaire algorithm. The Pyke Knowledge Engine was implemented using the Python Knowledge Engine (Pyke) library, and is itself composed of four components:
    - the Pyke Rule Engine runs the algorithm by using the Pyke's library rules engine;
    - the Pyke Fact Base, Pyke Rule Base and Pyke Question Base components are very similar, they are Pyke-specific files containing the facts, rules and questions that the Pyke Rule Engine will use for running the dynamic questionnaires.
- the Adapter component serves as a way for other systems to access the functionalities of this one. It is not yet implemented, as it is supposed to be created by the developers of the system that is meant to integrate with ours, according to their own needs.

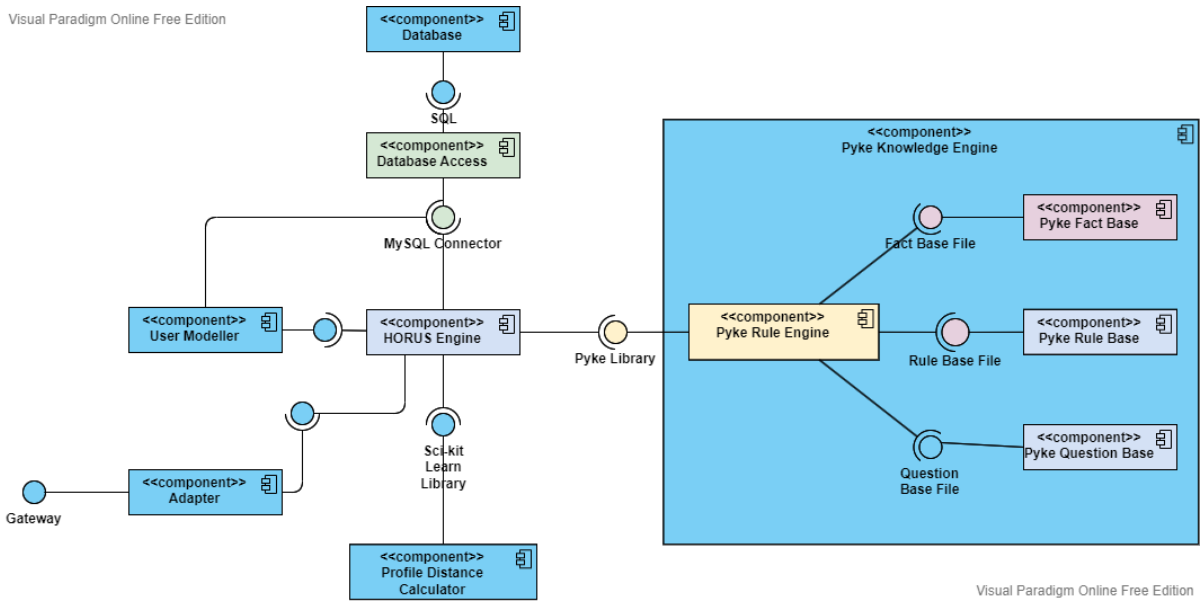

Figure 7 - System Component Diagram

3.4 Database

The database used in the system is a subset of the database used for the HORUS Project, containing only the tables that were needed inside the implemented functionalities' scope. HORUS' purpose is to improve Human Resources Management in organisations, by way of correctly evaluating their collaborators' knowledge and assigning them engineering profiles. Projects within the organisation will require positions to be filled by collaborators that have attained the specific engineering profile the position needs, and as such, having the collaborator's already been assigned engineering profiles makes it easier and less time consuming to fill those positions.

For an engineering profile to be attained, a collaborator needs to be knowledgeable in the skills expected of someone with that profile. How knowledgeable a collaborator is in a certain skill is measured using a metric denominated knowledge level; this metric is divided into four levels: Beginner, Intermediate, Advanced and Expert. Collaborators also have a seniority level within the organisation. As the collaborator's seniority level increases, the knowledge level in each skill required for an engineering profile also increases. As an example, the" C/C++ Embedded Engineer" profile requires a level of Intermediate in the skill "OO Programming" for the "Cons. Adv. Tech" seniority level, however, for the seniority level "Sr. Cons." a knowledge level of Expert in the same skill is needed.

In the same manner as engineering profiles, skill knowledge levels are attained by maintaining a certain knowledge in a skill's respective subtopics. Subtopics describe a specific subset of knowledge within a skill; for example, the skill "OO Programming" contains subtopics such as "Definition of classes and fields, "Basic design patterns" and "Use of generic classes". These subtopics are evaluated according to the first five levels of the Bloom taxonomy: Remember, Understand, Apply, Analyse, and Evaluate. Higher knowledge levels in skills require a higher level of the bloom taxonomy in the skill's subtopics. Lower knowledge levels might not even require any level of the bloom taxonomy in certain subtopics, as it is considered that, to attain that specific knowledge level in the skill, no knowledge of that specific subset of the skill is required, and that these subsets are better suited to more advanced knowledge levels. Bloom Taxonomy levels were assigned to every subtopic for each skill knowledge level by the HORUS project team.

Figure 8 depicts parts of the HORUS Project's database Entity-relationship Model where it is possible to more closely inspect these concepts and their relationships by seeing how they were implemented.

The tables Skill, SubTopic, Engineering Profile, and Collaborator are representations of the concepts they are named after, and as such require no further explanation.

The tables Knowledge Level Type, Bloom taxonomy Type and Profile Level Type represent the knowledge level of a skill, the bloom taxonomy level of a subtopic and the seniority level of a collaborator respectively. They contain only the description of the level and an integer representing the level itself.

The Collaborator Profile table represents an engineering profile that the collaborator was assigned to at some point. The “isPrimaryProfile” field contains a Boolean value that lets the system know if an engineering profile is the collaborator’s main profile. A collaborator can only be assigned one primary/main profile at a time. The “isHistoric” field also contains a Boolean value, however this value lets the system know if the collaborator still qualifies for an engineering profile.

The SubTopic Level table represents the Bloom taxonomy level needed in a certain subtopic for a knowledge level of a specific skill. The “Skill Level_ID” field references the Skill Level table which in turn represents a combination of a knowledge level and a skill. This reference in the SubTopic Level table is necessary because in this context subtopic/bloom taxonomy combinations exist only associated with specific skill/knowledge level combinations. These on the other hand, do not require association with other concepts such as engineering profiles, and as such, the Skill Level table does not contain a field referencing them.

The Profile Level Required Skill Level table represents a skill/knowledge level combination necessary for someone to possess an engineering profile in a specific seniority level.

The Collaborator Evaluation table represents an evaluation the collaborator underwent at some point, including their own self-evaluation. It also stores the date the evaluation took place.

The Collaborator Subtopics table represents the collaborators’ knowledge regarding every subtopic. Results from the evaluations are stored here, and as such the table contains the id of the evaluation from where each entry was inferred. It contains the id of the subtopic that was evaluated, and the resulting Bloom taxonomy level. The “isHistoric” field contains a Boolean value that informs if the entry is the most recent evaluation of that subtopic.

The Collaborator Skills table represents the collaborators’ knowledge regarding every skill. As such, it contains the id of the matching skill level. This table is automatically updated at the end of each evaluation, considering its results.

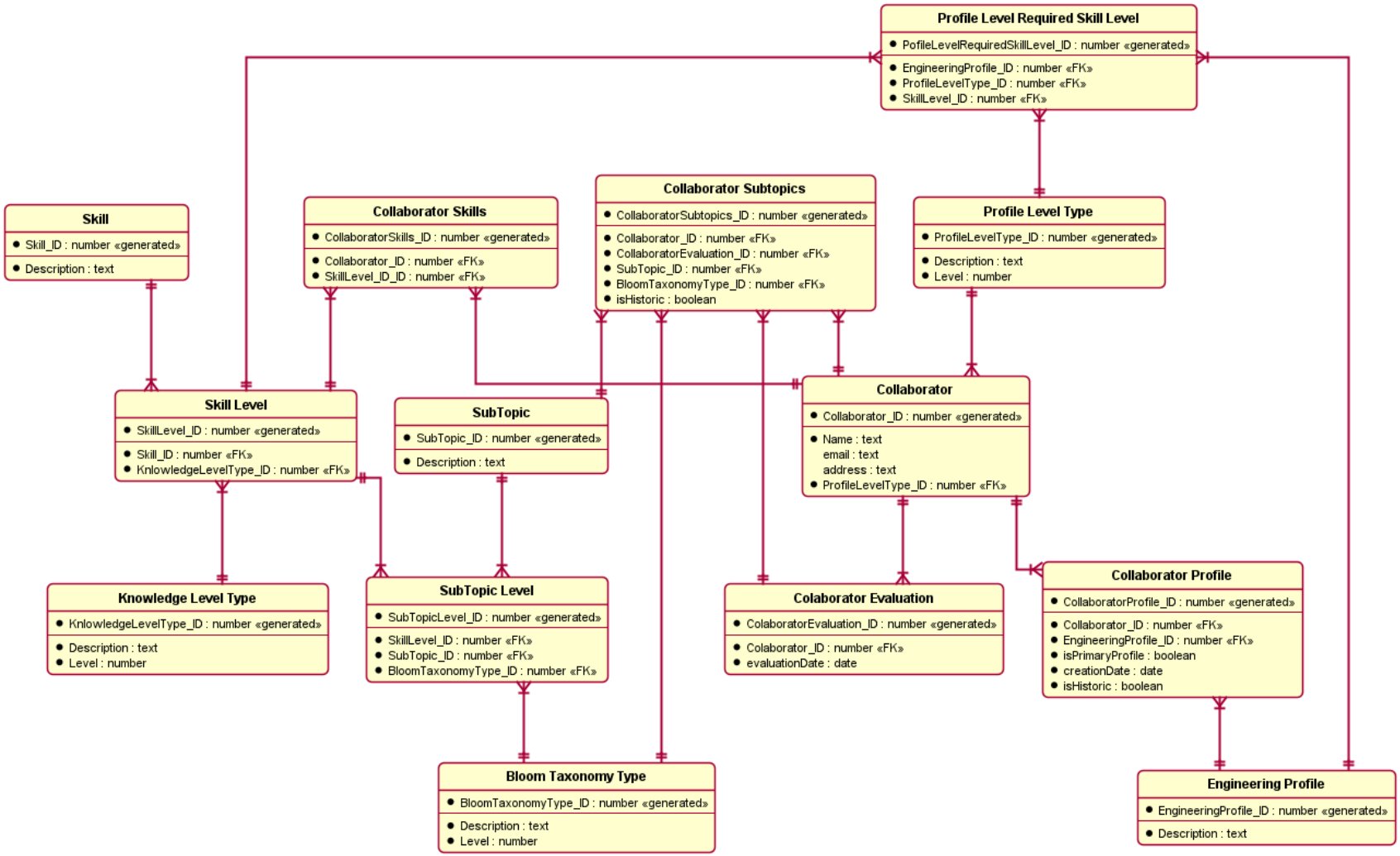

Figure 8 – Part of the HORUS Project's database Entity-Relationship Model

3.5 Database Access

This component uses MySQL Connector/Python [87], to connect to the database and perform any operations needed. By using an intermediary component between the HORUS Engine and User Modeller components and the Database, they both become independent of the implementation of the database, allowing for this component to be replaceable according to different database implementations.

This component starts by establishing a connection to the database when the system starts, as depicted in Code snippet 1.

```
connection = mysql.connector.connect(host='localhost',
                                     database='horus db',
                                     user='root',
                                     password='root')
```

Code snippet 1 - Establishing a connection to the database

After establishing a connection, the component creates a "cursor", which is a structure that handles any Data Definition Language statements. The creation is portrayed in Code snippet 2.

```
cursor = connection.cursor()
```

Code snippet 2 - Cursor structure creation

Using the cursor, it is now possible to perform any operation related to the database. Any operation is implemented in their own methods, with names as generic as possible. Since the other components call the methods by name, as long as the method's name is the same, and return values are structured similarly, the implementation of this component can be changed to suit the database's implementation. Code snippet 3 depicts a method used to retrieve a collaborator's current bloom taxonomy level in a specific subtopic.

```
def getCollaboratorCurrentSubtopicKnowledge(collaborator, subtopic):
    sql_select_query = f"""SELECT bt.Level
                           FROM `horus db`.collaborator as col,
                           `horus db`.`collaborator subtopics` as colSub,
                           `horus db`.subtopic as sub, `horus db`.`bloom taxonomy` as bt
                           WHERE colSub.Collaborator_ID = col.ID AND colSub.isHistoric = 0
                           AND colSub.Subtopic_ID = sub.ID
                           AND bt.ID = colSub.`Bloom Taxonomy_ID`
                           AND sub.Description = \"{subtopic}\"
                           AND col.Name = \"{collaborator}\";"""
    cursor.execute(sql_select_query)
    try:
        return cursor.fetchone()[0];
    except Exception:
        return 0
```

Code snippet 3 - Fetching a collaborator's bloom taxonomy level in a subtopic

The method receives as arguments the names of both the collaborator that possesses the knowledge and the subtopic that we want to retrieve the bloom taxonomy level. We start by creating a query “sql_select_query” which contains the Data Definition Language statement to be executed in the form of a string. The cursor then executes the query and the database readies itself to return the data it gathered. This data can be comprised of any number of rows, however, as the collaborator only has one bloom taxonomy value that fits the query, we use the method “cursor.fetchone()” to retrieve the only row returned by the database. If the collaborator does not yet have any knowledge in the specified subtopic, the method returns the value “0”.

3.6 HORUS Engine

The HORUS Engine component serves as a middleware that links all other components together and provides a gateway for other systems to use the implemented functionalities. It is the component that dictates the flow of the program and utilises every other component so it can use their functionalities to perform its function. It has access to the database to fetch or update any data needed that does not relate to the user model, as that is the User Modeller component’s responsibility.

Figure 9 contains a flowchart that depicts how the HORUS Engine component works.

The system starts by presenting a list of names of collaborators that it fetched from the database and tells the user to pick one. This part functions as a placeholder, as the full system when integrated with Capgemini’s Internal system will have the collaborators authenticate themselves instead.

Then, the user is asked for their seniority level in the organisation. According to the indication of the user, the system infers the knowledge levels of the collaborator in each skill, using the entries in the Profile Level Required Skill Level table in the database. The user model is then created by the User Modeller component based on the inferred knowledge levels. This entire section of the program also functions as a placeholder, as in the full system, the knowledge levels will be fetched from a previously conducted self-evaluation.

From this point the system lets the user choose which functionality use: they can either calculate the engineering profile they are closer to attaining or evaluate their knowledge in a specific skill. Code snippet 4 shows the implementation of the system printing to the console the possible functionalities that the user can pick. Note that in the full system, the first time the user logged-in to the system they would be asked to perform a self-evaluation of their knowledge before being able to access any of these functionalities, as to let the system create their user model.

```
while True:
    print("1 - Find closest profile")
    print("2 - Test collaborator knowledge")
    word = input("Decide what to do or type \"end\" to exit:")
    if word == "end":
        break
```

Code snippet 4 - Letting the user pick a functionality to use

Lastly, the system will execute the chosen functionality and make changes to the user model according to the results.

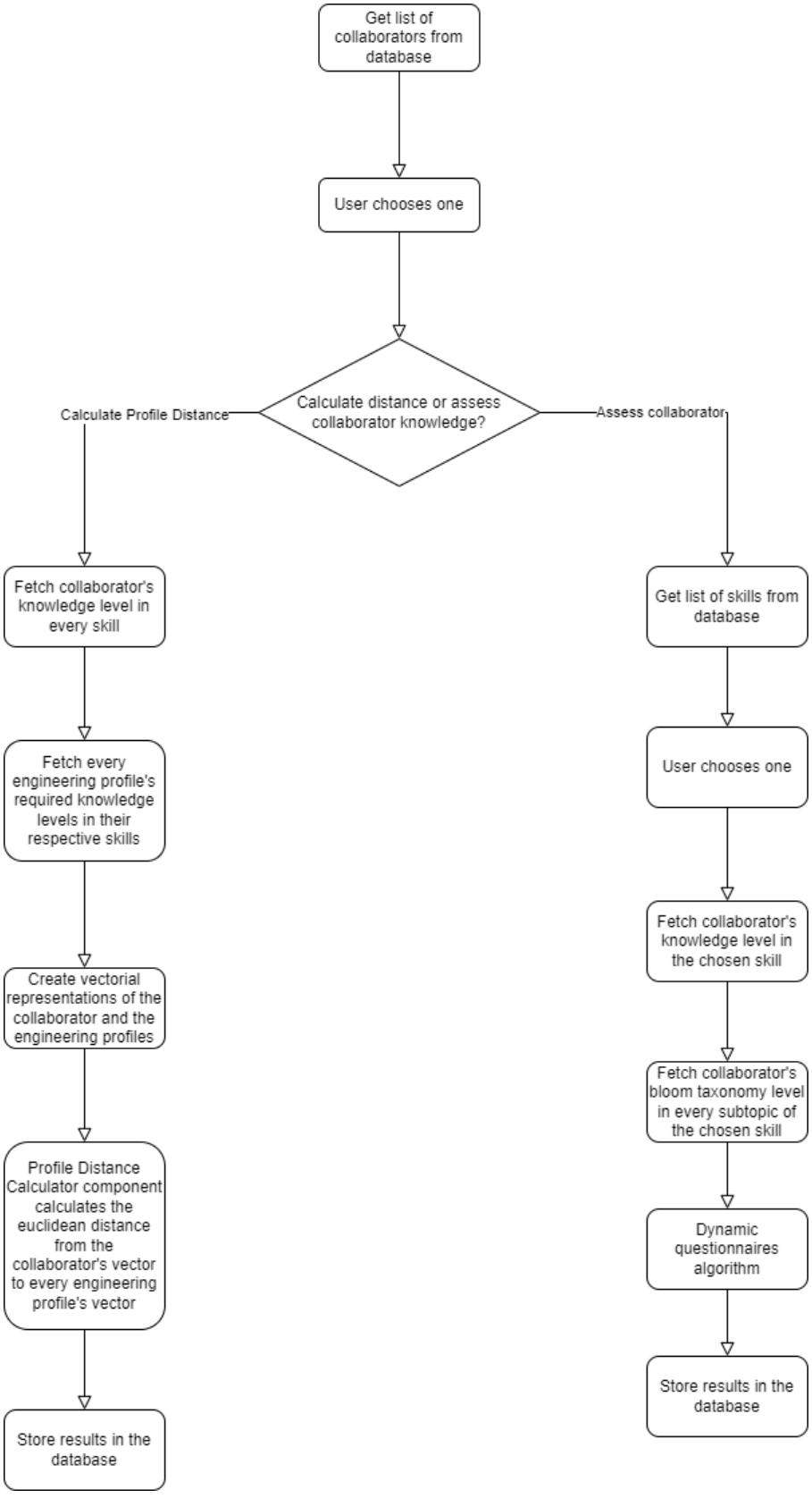

Figure 9 – HORUS Engine flowchart

The Pyke Knowledge Engine is the component is utilised by the HORUS Engine to obtain the current knowledge level in a certain skill. The data returned will be composed of the bloom taxonomy levels in each subtopic the user was evaluated in and the matching knowledge level in the evaluated skill. The HORUS Engine then tells the User Modeller to update the user model in the database with the new values for each subtopic and skill.

The Profile Distance Calculator component is utilised by the HORUS Engine when the need arises to know which engineering profile the collaborator is closest to attaining. The data returned is composed of a list of the existing engineering profiles ordered in ascending order of the distance between the collaborator's knowledge and the engineering profile. This data is then used by the User Modeller to update the user model.

The User Modeller component is utilised by the HORUS Engine whenever any operation needs to be done regarding the user model. This includes creating it, fetching information from it, and updating it.

3.7 User Modeller

The User Modeller constructs the user models of collaborators and is responsible for any updates or changes to them.

The user is represented in a similar manner to the knowledge stored in the database. Being mainly portrayed by a set of the collaborator's engineering profiles, one of which being their primary profile, while all others being secondary. The user models' engineering profiles are then divided into their respective skills with the user's corresponding knowledge levels in each, going from "Beginner" to "Expert". Finally, the user model's skills are divided into subtopics, each of them matched with the user's bloom taxonomy levels in that subtopic, going from "Remember" to "Evaluate". As such, the User Modeller uses an Overlay Model approach to build the user model, where each knowledge level and bloom taxonomy level is only a fraction of the domain's existing knowledge in each skill and subtopic.

Figure 10 depicts the flow of the User Modeller within the larger system.

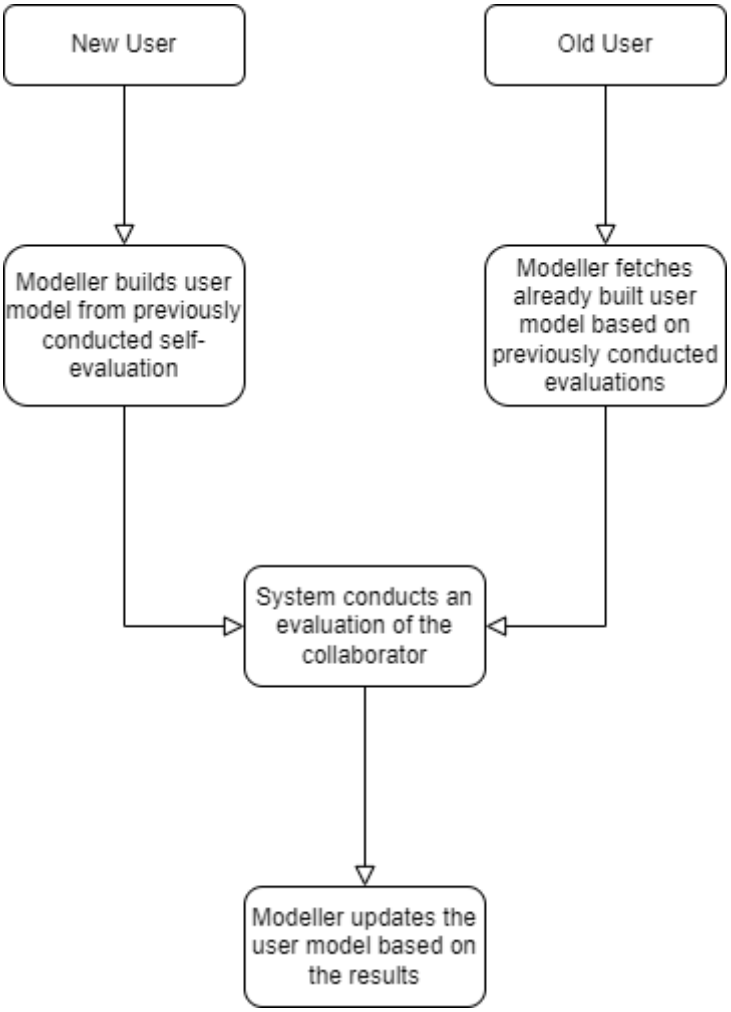

Figure 10 - Flow of the User Modeller

A new user is always asked to perform a self-evaluation, in which they will provide values (from one to five) for the bloom taxonomy level they think they have in every subtopic. These values are stored by the User Modeller in the Collaborator Evaluation and Collaborator Subtopic tables in the database. The User Modeller also uses the values to calculate the knowledge levels the collaborator has in each skill and stores them in the Collaborator Skills table. Note that the self-assessment functionality is implemented in Capgemini's Internal System and is not part of this system. If the user is not new however, the system builds the user model based on the information already contained in the database. This data relates to the other previously conducted evaluations of the collaborator, and their results.

The User Modeller updates the user model during and after the conclusion of each evaluation. Every time a subtopic is evaluated, the user model must be updated with the resulting Bloom taxonomy level. This level is a new entry in the Collaborator Subtopics database. Previous entries for the same subtopic are then changed to historic, by changing the value of the "isHistoric" field and are no longer relevant to the user model. At the end of the evaluation, the modeller uses the results of the recently conducted evaluation to calculate the new knowledge

levels for every skill and updates the user model with the respective values. Even though only one skill is chosen to conduct an evaluation, since the same subtopic can be part of multiple skills, every skill's knowledge level must be recalculated. This behaviour is presented in Code snippet 5. The component also behaves in a similar manner during and upon conclusion of a self-evaluation.

```
def updateUserModel(collaboratorName, subtopicKnowledgeDict):
    skillKnowledgeDict = calculateSkillKnowledgeLevels(subtopicKnowledgeDict)
    for skill in skillKnowledgeDict:
        DataAccess.updateCollaboratorSkill(collaboratorName, skill, skillKnowledgeDict[skill])
    return skillKnowledgeDict
```

Code snippet 5 - Updating the user model

The method "updateUserModel" functions as described previously. It receives the name of the collaborator that conducted the evaluation, and the results of the evaluation in the form of a Python dictionary. The dictionary's keys are the subtopics evaluated, and its values the are resulting Bloom taxonomy levels from one through five. The modeller starts by calculating the knowledge level of every skill based on the results of the subtopics evaluation. It the asks the Database Access component to update the collaborator's skills in the database with the calculated values. Lastly it returns the same values for them to be available to be displayed to the user.

3.8 Profile Distance Calculator

This component has the functionality of identifying which of the engineering profiles collaborators are closest to attaining. This serves two main purposes: the first is to provide a goal for collaborators to strive towards, boosting their motivation and bolstering their knowledge in the process of trying to achieve it; the second is to more easily fill roles that require a specific engineering profile. If a project has a position that requires a particular engineering profile but there are no available collaborators with the qualifications to fill it, it is possible to use this functionality to find a collaborator that has almost all the qualifications necessary and requires the minimum training to be able to fill the position.

Both the skills required to obtain attain an engineering profile and the skills a collaborator already possesses are ranked in knowledge level according to a scale from one through four, where each of the numbers represent, in order, Beginner, Intermediate, Advanced and Expert knowledge levels. As such, it was decided that engineering profiles and collaborators could be compared by depicting them using n-dimensional vectors, n representing the number of skills an engineering profile requires. In Figure 11 it is possible to see an excerpt of the program's output when comparing the collaborator's knowledge to all seniority levels of the engineering profile "C/C++ Embedded Engineer".

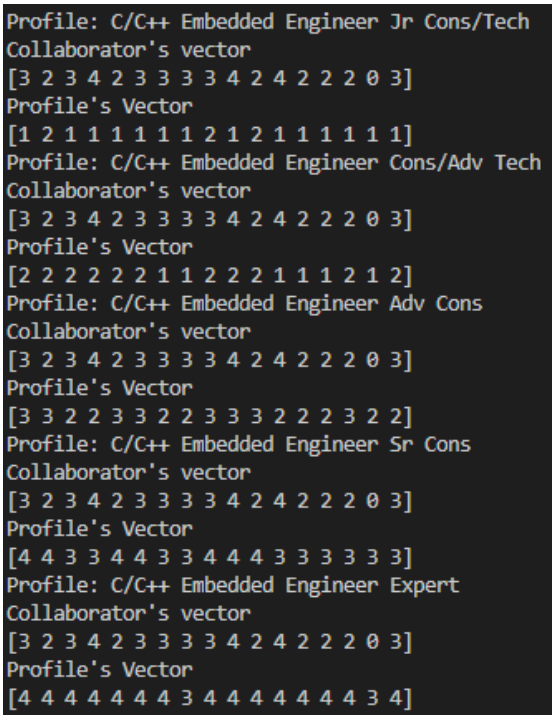

Figure 11 - Excerpt of the program's output when comparing collaborator's knowledge to an engineering profile

As previously alluded to, each of the values in the vectors represents the knowledge level in a skill necessary to attain the engineering profile the collaborator is being compared to. These values go from one through four in accordance with the existing knowledge levels. The zero value in a collaborator's vector represents that no knowledge was held about that specific skill.

Comparing two n-dimensional vectors is a simple ordeal and can be done in multiple ways. The Euclidean distance was chosen to perform this task, as it uses the Pythagorean Theorem to calculate distances between points in an n-dimensional space based on their Cartesian coordinates. The Sci-kit Learn library [88] was used for this purpose, as the author was already very familiarised with it and needed no further training on how to use it. The implementation is depicted in Code snippet 6, where the sci-kit learn's Euclidean distance method receives the vectorial representation of the collaborator and profile to be compared as an argument, and returns the calculated value.

```
return euclidean_distances([collabv,profilev])
```

Code snippet 6 - Sci-kit learn's euclidean distance method

After calculating all the distances between the collaborator and every engineering profile and seniority level combination, the profile that resulted in the lowest distance calculated is considered to be the one the collaborator is closest to. Note that in the example seen in the previous figure, the collaborator had no profile assigned and as such, every profile and seniority level combination was considered. However, if the collaborator already has engineering profiles

assigned, the system will only consider the profiles of the same seniority level (if the collaborator doesn't already possess them) or of a higher level of seniority than the ones the collaborator already possesses.

3.9 Pyke Knowledge Engine

The Pyke Knowledge Engine component was implemented using the Python library Python Knowledge Engine (Pyke). Pyke provides a knowledge-based inference engine and a way to use Logic Programming in python. This is done by writing files of various Pyke-specific extensions in a certain syntax. These files will then be analysed and converted by Pyke into a fully functioning python program.

According to the characteristics listed in 2.1.2, Pyke was chosen because it allows for the implementation to be fully in python, therefore not being necessary to create/change the environment that is already being used by the rest of the system, it supports both forward-chaining and backwards-chaining inferencing, and most importantly, it has a dedicated functionality to work questions into the rules engine, thus aligning perfectly with the project's goals.

3.9.1 Pyke Fact Base

A fact is a data value that Pyke acts upon. Facts are true statements and represented in fact bases in a form that could be interpreted as a condensed form of a spoken sentence. As an example, the sentence "Collaborator Lucy's primary engineering profile is C/C++ Embedded Engineer" could be condensed and stored in a fact base in the form "primary_profile(Lucy, Embedded Engineer)".

The knowledge engine's fact base starts out with very few facts, containing only a depiction of the Bloom taxonomy in the form of "bloom_taxonomy(*Level in text form*, *Level in number form*)".

When it is chosen a collaborator to take the questionnaire, and a skill for the questionnaire to be about, the system will fetch all the subtopics of that skill from the database and assert them as facts in the fact base in the form of "subtopic(*Subtopic name*)".

Pyke fact bases are stored in ".kfb" files which can be created and edited with any text editor. Pyke will try to match facts in specified fact bases to the rules called by the knowledge engine. Code snippet 7 depicts how the bloom taxonomy is established as facts in a ".kfb" file.

```
bloom_taxonomy(Remember,1)
bloom_taxonomy(Understand,2)
bloom_taxonomy(Apply,3)
bloom_taxonomy(Analyse,4)
bloom_taxonomy(Evaluate,5)
```

Code snippet 7 - Facts in a ".kfb" file

3.9.2 Pyke Rule Base

In Logic Programming, rules take the form of "if A then B", both A and B being statements. A is the premise statement and if it is true, then B (the conclusion statement) is inferred to be true. Since premises within the if-clause are general statements, they are matched with the known facts in the fact base, which can result in a match with more than one fact. Backtracking is a process that allows Pyke to try all the combinations of matching facts to successfully prove the premise. This can also cause a rule to succeed more than once for multiple unique combinations of facts. Figure 12 depicts how the backtracking functions when trying to prove the premises A, B and C of a simple rule.

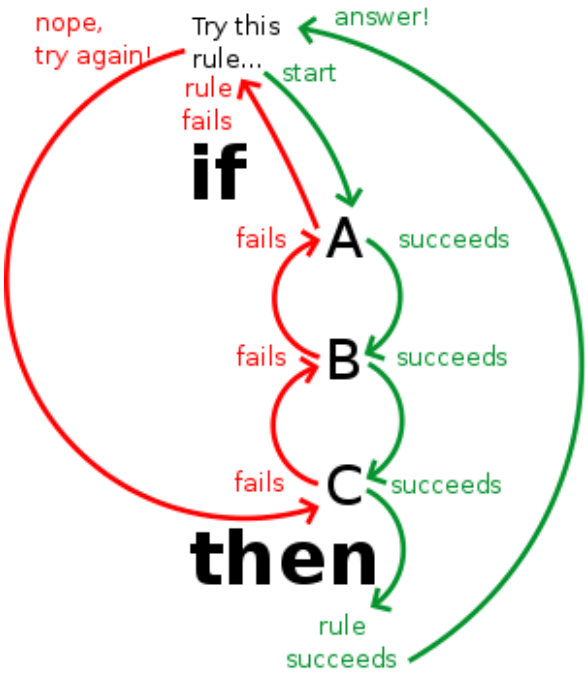

Figure 12 – Pyke backtracking example [89]

The Pyke knowledge engine's rule bases (collections of rules) are contained in ".krb" files which can be created and edited with any text editor. Rule bases can contain both forward-chaining rules and backward-chaining rules.

Forward-chaining rules are used to assert facts if their premise is deemed to be true (through fact matching and backtracking explained previously in this section). All forward-chaining rules are run whenever the rule base that they are a part of is activated. An example of this type of rule can be seen in Code snippet 8.

```
subtopics_to_evaluate_rule
    foreach
        factbase.subtopic($subtopic)
        factbase.bloom_taxonomy($description,$level)
    assert
        factbase.subtopic_level($subtopic,$level)
```

Code snippet 8 – Forward-chaining rule example

The example code starts with the name of the rule "subtopics_to_evaluate_rule". This is then followed by the rule's premise : the "foreach" keyword serves the purpose of the "if" clause, "factbase.subtopic($subtopic)" means that the program will try to match the premise to any "subtopic" fact with the respective value (the variable "$subtopic") it finds in the fact base "factbase"; "factbase.bloom_taxonomy($descripton,$level)" works in a similar manner. For every unique combination of matching facts it encounters the program will then execute the conclusion premise (the "assert" keyword represents the "then" clause in a rule), which in this case is to assert a "subtopic_level" fact in the fact base "factbase" with the values of "$subtopic" and "$level" from the matching premise facts.

Backwards-chaining rules are used when a question is asked of the knowledge engine, meaning when it is asked to prove a specific goal. Thus, Pyke starts by finding the rules whose "then" part matches the goal that it was asked to prove, and then tries to prove all of the subgoals in the "if" part of the same rules. In essence subgoals will help satisfy some part of the original goal and can be divided into other subgoals. As such, subgoals can be both matched against facts in a knowledge base or against other backward-chaining rules in the same rule base. The original goal is proven if all subgoals can also be proven. An example of this type of rule can be seen in Code snippet 9.

```
check_difficulty_level_increase
    use check_result($correctanswers,$incorrectanswers,$level,$newlevel)
    when
        check $correctanswers = 2
        python $newlevel = $level + 1
```

Code snippet 9 - Backward-chaining rule example

Pyke's backward chaining rules use the keywords "use" and "when" in place of the "then" and "if" clauses respectively. The first line contains the name of the rule, "check_difficulty_level_increase". Followed by the "use" ("then") part of the rule which Pyke will attempt to match to either the goal it was asked to prove, or the rule called by the previous rule. This part of the rule is inferred to be true if the "when" ("if") part of the rule is also inferred

to be true. The parameters “$correctanswers” and “$incorrectanswers” are the number of correct and incorrect answers the user has given, the “$level” parameter is the current difficulty level of the questions, lastly, the “$newlevel” parameter will return the new level of difficulty to the previous rule. The “check” keyword will check if a condition is met, in this case, if the number of correct answers is two. The “python” keyword allows the rule to run python code, in this case, adding one to the current level of difficulty and assigning the value to “$newlevel”.

3.9.3 Pyke Question Base

The questions used by the system should evaluate parts of a certain skill, as to allow the questionnaires to use multiples of them to get an accurate picture of a collaborator’s knowledge regarding the same skill.

As skills are divided into subtopics, in accordance with 3.4, and each knowledge level of a skill has a bloom taxonomy level assigned to each subtopic that composes it, the questions were designed so that each one evaluates the level of the bloom taxonomy a collaborator possesses in one or more subtopics. Since each question evaluates one or more subtopics, by using multiple questions, it is possible to get a complete picture of what the knowledge of the evaluated collaborator looks like.

The questions were obtained from two sources: either from the question repository from Capgemini Engineering, or created by the HORUS project team. All questions are in the form of a multiple-choice question with three possible incorrect answers and one correct answer. A level of the bloom taxonomy is assigned to each subtopic a question is evaluating. As an example, the question “Which type of members can’t be accessed in derived classes of a base class?”, has the wrong answers: “All can be accessed”, “Protected” and “Public”; and the one correct answer: “Private”. The same question evaluates three subtopics: “Class-hierarchy design for modelling” with a bloom taxonomy level of “Understand”, “Definition of classes and fields” with a bloom taxonomy level of “Apply” and “Privacy and visibility of class members” with a bloom taxonomy level of “Understand.

The main feature that contributed to Pyke being chosen to develop this system was that it could ask the end users questions. Questions can be called either directly by the engine or by rules and are stored in “.kqb” files. Code snippet 10 demonstrates an example of the implementation of a question.

```
question(Basic and structured data types,2,$ans)
    Which of these is not a structured data type?
    ---
    $ans = select_1
        1: array
        2: struct
        3: union
        4: char
```

Code snippet 10 - Example of a Pyke question

Every question in the question base starts with the keyword “question” followed by some parameters. In this case the first two parameters are used so that when a rule calls this question in the form of “question($subtopic, $level, $ans)”, this question is only asked if the values of “$subtopic” and “$level” match with “Basic and structured data types” and “2”. Below the parameters is the question body, followed by a separator “---” and the type of the question and possible answers. The “$ans” parameter will contain the answer given by the user, which, since the question type is “select_1” must be one of the four possible answers “array”, “struct”, “union” or “char”. When a rule calls a question, it will receive the answer of the user and can then perform any check it needs to fulfil its premise. As an example, a rule that calls this question could check if the answer is correct by examining if the answer given was “char”.

3.9.4 Pyke Rule Engine

Pyke’s rule engine works mostly using an “engine” object, specific to the library. An engine can manage multiple knowledge bases, meaning fact, rule and question bases. After the creation of an engine, Pyke scans for knowledge base files (“.kfb”, “.krb” and “.kqb” extensions) and compiles them into python code that can be ran later. Code snippet 11 depicts the creation of an engine object. The main argument for the creation of the engine is the path to the directory containing the files to compile into python code; since the files are in the same directory, “__file__” is passed as the value.

```
from pyke import knowledge_engine
engine = knowledge_engine.engine(__file__)
```

Code snippet 11 - Creating an engine object

The second step is for the engine to assert facts upon the fact base using the function “add_universal_fact”. Unlike facts already written in the “.kfb” files, these facts are temporary and can then be deleted when performing a reset of the engine. Code snippet 12 depicts a “subtopic” fact being asserted by the engine, done after a collaborator and skill are chosen, according to what was mentioned in 3.9.1. The tuple contains the arguments of the fact, in this case a single argument with the name of the subtopic “Basic and structured data types”.

```
engine.add_universal_fact('subtopic',('Basic and structured data types'))
```

Code snippet 12 - A fact being asserted by the engine

After the creation of the engine and assertion of facts, a rule base must be activated, so as to associate it with the engine. Forward-chaining rules are activated at this time. The activation is done using the “activate(rule base name)” function as depicted in Code snippet 13 with the engine object activating the rule base with the name “rulebase”.

```
engine.activate('rulebase')
```

Code snippet 13 - Rule base activation

The last step of working with the knowledge engine is to ask it to prove a goal. As Pyke can return multiple answers to a single goal, we can use the “prove_1_goal(goal, **args)” function to return only the first answer, or the “prove_goal(goal, **args)” function to get all of the answers. Code snippet 14 shows a call for the engine to prove a goal by inferring “test_collaborator” with the parameters “$collaborator” taking the value of “Lucy and “$skill” taking the value of “OO Programming”. Proof of this goal is returned as a tuple containing the values for the parameters that were not assigned one, in this case the tuple will contain the value of the variable “$result”.

```
engine.prove_1_goal('rulebase.test_collaborator($collaborator,$skill,$result)',
                    collaborator = "Lucy",
                    skill = "OO Programming")
```

Code snippet 14 - Proving a goal

3.9.5 Dynamic Questionnaires Algorithm

The questionnaires are performed by a collaborator in relation to a specific skill that needs to be evaluated to determine the collaborator’s knowledge level of that skill at the point in time the evaluation is carried out, and if it meets the objective. The dynamic nature of the questionnaires meant that they would need to change, adapting to the user’s answers to the questions. This dynamic nature helps in providing a more correct evaluation of the knowledge the user holds. The questionnaire starts when the user selects one of the skills shown on screen to be evaluated, as shown in Figure 13.

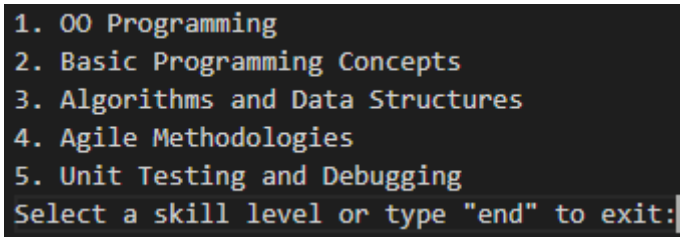

Figure 13 - Choosing a skill to evaluate

After choosing a skill, the questionnaire picks a subtopic at random to start the evaluation. Afterwards, the system fetches the bloom taxonomy level the collaborator has in that subtopic, and all questions that evaluate that subtopic for that bloom taxonomy level. The algorithm will then start to perform the evaluation of that subtopic, starting with a question corresponding to the collaborator's bloom taxonomy level, as seen in Figure 14, and behaving as depicted in the flowchart in Figure 15. The current bloom taxonomy level being evaluated is also called the question difficulty level.

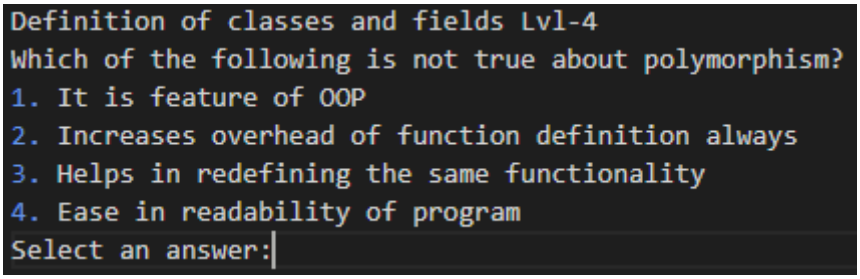

Figure 14 - Example of a question being presented by the system

After receiving an answer for the question presented to the user the system checks if the answer is correct or not. Since the questions are multiple-choice, answering a question correctly doesn't immediately mean the user has the required knowledge, it could have been a lucky guess. Similarly, if the user gets a question wrong, it could be attributed to a singular mistake in reasoning and doesn't mean the user does not have knowledge of the subtopics. Thus, a threshold had to be established. Users whose percentage of correct answers is equal or higher than the threshold are considered to possess the knowledge necessary about that subtopic and move on to questions of a higher difficulty level. Users who can't match the threshold are considered to not possess enough knowledge about that subtopic and the difficulty level of the questions being asked is lowered. Considering the time required to perform a test, and in an effort to lower it as much as possible, the system is limited to asking three questions per subtopic and difficulty level combination. As such, the threshold was defined as 66%, meaning that users who answer correctly in two out of the three questions pass the current difficulty level, and users who answer two questions incorrectly fail and the difficulty level is lowered.

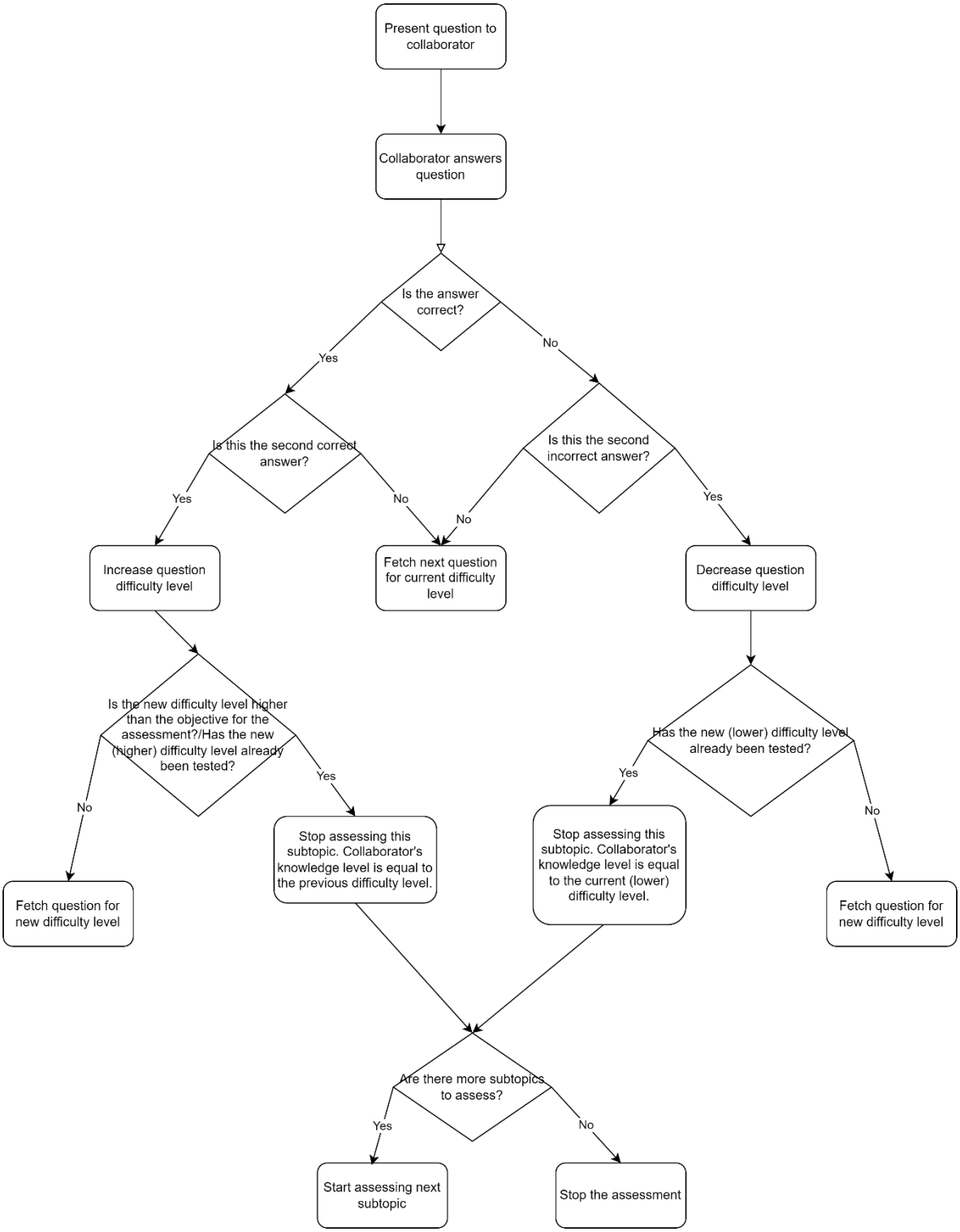

Figure 15 - Dynamic questionnaire algorithm

If the answer given is not enough to change difficulty level, the system simply moves on to the next question in the current difficulty level.

When increasing difficulty level there are two conditions that, if met, stop the evaluation of the current subtopic.

If the new difficulty level is above the objective of the evaluation, it means the collaborator passed the difficulty level required by the organisation. An objective for the questionnaire is usually a skill's knowledge level that the collaborator is expected to achieve. As an example, if the collaborator currently is an Intermediate level in a skill, but the organisation requires the collaborator to fill a position that needs someone with an Advanced level in the same skill, this will be treated as the objective of the questionnaire. The questionnaire stops evaluating a subtopic when the objective bloom taxonomy level is met as to further reduce the time necessary to perform the evaluation.

On the condition that the new difficulty level has been evaluated previously, meaning: the collaborator failed it, the difficulty level was lowered, and in this lowered level the collaborator answered correctly twice, increasing the difficulty to the already failed difficulty level. This second condition also stops the evaluation of the current subtopic, as it is not desirable to once more test something that the collaborator already failed.

When lowering difficulty level there is one condition that if met, stops the evaluation of the current subtopic.

Provided that the new difficulty level has already been evaluated once. Similar to the previously explained condition, if the collaborator passes a difficulty level, but fails the next level and goes back to the first, there is no need to test what has already been tested.

When the collaborator fails when the difficulty level is at its lowest, "Beginner" in the bloom taxonomy, or when the collaborator passes the highest difficulty level, "Evaluate" in the bloom taxonomy, the evaluation of the subtopic is stopped, and the collaborator gets a result of zero or five accordingly.

After checking if any of these conditions were met, the system acts accordingly. If the difficulty level is changed, either increased or lowered, the system fetches questions of the new difficulty level to present to the user. If the evaluation of the current subtopic is stopped, the system registers the result, the bloom taxonomy level equivalent to the difficulty level, and this becomes the new level of the collaborator in the subtopic.

Following a stop to an evaluation of a subtopic, the system will check if there are any more subtopics left to evaluate. If the evaluation is to continue in the face of the existence of more subtopics to evaluate, the system gives priority to subtopics that already have at least one answer.

Since a question can correspond to more than one subtopic, after the evaluation of a subtopic, depending on the questions presented to the user, there could be more subtopics that have

either, already been fully evaluated, or are partially evaluated. As an example, if the question "Which type of members can't be accessed in derived classes of a base class?" described in 3.9.3 is presented to the user and is answered correctly, all three of its subtopics progress in their evaluation, so even when the evaluation of the current subtopic is completed, the other two are already in progress and are thus given priority when choosing the next subtopic to evaluate.

To further reduce the time it takes for the evaluations to be completed, the system prioritises presenting questions that have more than one subtopic without ever presenting more than the necessary number of questions to evaluate the same subtopics. Meaning that if a subtopic was already fully evaluated, even if it was never the main subtopic being evaluated, the system will not present the user with any more questions regarding it. This helps to evaluate more subtopics in less time.

When every subtopic has been properly evaluated, the user is presented the results of their evaluation. The output first details the starting knowledge of the collaborator in each subtopic, following it with the results of the evaluation, where the user can see their final level in each subtopic, and lastly presenting the objective of the evaluation with the corresponding bloom taxonomy levels the collaborator was supposed to achieve. This output can be seen in Figure 16, in correspondence with what was just described, one can see the collaborator's starting knowledge, new knowledge (evaluation results) and goal knowledge (objective).

```
Collaborator's starting knowledge
{'Basic and structured data types': 3, 'Decomposition into objects carrying state
and having behaviour': 3, 'Class-hierarchy design for modelling': 2, 'Definition
of classes and fields': 0 'Definition of methods': 3, 'Definition of
Constructors': 4}

Collaborator's new knowledge
{'Basic and structured data types': 2, 'Decomposition into objects carrying state
and having behaviour': 2, 'Class-hierarchy design for modelling': 1, 'Definition
of classes and fields': 4, 'Definition of methods': 4, 'Definition of
Constructors': 2}

Collaborator's goal knowledge
{'Basic and structured data types': 4, 'Decomposition into objects carrying state
and having behaviour': 4, 'Class-hierarchy design for modelling': 4, 'Definition
of classes and fields': 4, 'Definition of methods': 4, 'Definition of
Constructors': 4}
```

Figure 16 - Output of the evaluation of a skill with only six subtopics created for demonstration purposes

3.10 Adapter

The adapter component is not implemented. It is supposed to be created by the developers of the system that wants to integrate with ours, according to their own needs. In developing the adapter, one must follow three requirements.

The first requirement is that it must implement an interface that connects their system to ours. This can be implemented using their preferred technology and techniques, however it must provide the information detailed in the other two requirements.

The second requirement for this adapter is that it must provide the identity of the user that will use the system. In the case of Capgemini Engineering's internal system, it is capable of authenticating the user. The adapter can then pass the relevant user information to the HORUS engine.

The third requirement is that it must provide the necessary knowledge for the user model to be created for each new user of the system. Capgemini Engineering's internal system has a self-evaluation module. The results of this self-evaluation can then be passed to the HORUS engine, which uses them to create the user model.

Hopefully these three requirements serve as a guideline when implementing the Adapter component.

4 Experimentation and Results

This chapter presents the assessment and experiments done for the system developed, presented in 3. Starting with what was assessed, following with how the assessment was performed and lastly, presenting the results of the assessment.

4.1 Assessment Objectives

The system developed has the main purpose of evaluating a collaborator's knowledge using dynamic questionnaires that adapt the questions presented according to how the user answers. As such, the main objectives of the system assessment were:

- Does the system effectively present a different questionnaire for every collaborator that used it?
- Are the questionnaires able to be completed within a specific time frame?
- Are the conclusions drawn by the system similar to the expected knowledge of a collaborator?

4.2 System Experimentation

Capgemini Engineering collaborators were selected, using Convenience Sampling, to participate in the testing of the system. Convenience Sampling was used due to the short time remaining for the system to be tested. Considering the time constraint, the need arose to find as many participants available in the shortest time possible.

In order to perform this assessment, eleven sampled collaborators attempted to complete a questionnaire given by the system. The participants had different seniority levels within the organisation, which gave us as ample and varied of a sample size as possible and allowed for the system to be tested from different starting knowledge levels.

This experimentation worked under two assumptions:

- The questions created for this purpose were treated as if created by experts and therefore could be used as a competent form of evaluation. This assumption is made due to the absence of the question crowdsourcing component of the system described in 1.1;
- The starting knowledge level for every skill of each collaborator was the level demanded by their seniority level in the organisation. This assumption was made due to the absence of self-evaluations to be used as the starting knowledge of the collaborators.

Object Oriented Programming (OOP) was the skill chosen to be evaluated, as there is a larger population of Capgemini collaborators that possess some level of knowledge in it. It also had the largest number of questions already created.

As we did not have access to a self-evaluation for each collaborator that tested the system, the starting knowledge level of each participant was defined in accordance with their seniority level. The objective knowledge level was set one level above the starting knowledge level. As an example, a collaborator with a seniority level of "Adv. Cons." would have a starting knowledge level in OOP of "Advanced", and an objective/goal knowledge level of "Expert". Seniority levels are matched with each skill's knowledge level in the database, and these relationships were defined by the HORUS project team. Table 4 depicts how the OOP knowledge levels relate to the seniority levels.

Table 4 - Knowledge Levels in OOP according to seniority level

Skill	Jr Cons./Tech	Cons./Adv. Tech	Adv. Cons.	Sr. Cons.	Expert
OOP	1	2	3	4	4

Every participant had a session with a set time frame of thirty minutes to try to complete the questionnaire. Some participants, according to their availability, were allowed to keep answering after the thirty minutes had passed, until they finished the questionnaire. This allowed for testing how much time is needed to complete the questionnaires in their current form.

As higher levels of knowledge in a skill incorporate more subtopics than lower knowledge levels, participants of a higher seniority level, starting with a higher knowledge level, had more subtopics to be tested, and therefore their questionnaires were expected to last longer than those with a lower seniority level.

The questions used by the system for this test were created specifically for this purpose by the HORUS project team and will not be the used by the final system.

After a first round of experimentation with the eleven Capgemini Engineering collaborators, an oversight in the implementation was discovered. The system could not stop an evaluation

midway through it, and as such, the results of evaluations that were interrupted were unobtainable. As such, a second round of experimentation was done with three more collaborators and an implementation that fixed this oversight.

Table 5 depicts the distribution of the fourteen participants across the seniority levels in Capgemini Engineering.

Table 5 – Distribution of participants per seniority level

	Jr Cons./Tech	Cons./Adv. Tech.	Adv. Cons.	Sr. Cons.	Expert
Number of Collaborators	4	2	4	3	1

Considering the convenience sampling method used, the distribution of the collaborators per seniority level is quite varied. The "Jr Cons./Tech" and "Adv. Cons." levels have the most representation, with four participants each. The "Expert" level is the least represented, with only one participant.

4.3 Assessment Results

During both rounds of experimentations, the time taken for each participant to complete the questionnaire was registered, as well as the results of the questionnaire itself. The time to complete the questionnaire is depicted in Table 6.

Table 6 - Time to complete the questionnaire

Collaborator	Seniority Level	Time to complete
Collaborator 1	Sr. Cons.	Not completed
Collaborator 2	Adv. Cons.	Not completed
Collaborator 3	Jr. Cons./Tech	Not completed
Collaborator 4	Cons./Adv. Tech	52 minutes
Collaborator 5	Adv. Cons.	Not completed
Collaborator 6	Expert	Not completed
Collaborator 7	Adv. Cons.	Not completed
Collaborator 8	Sr. Cons.	36 minutes
Collaborator 9	Cons./Adv. Tech	60 minutes
Collaborator 10	Jr Cons./ Tech	35 minutes
Collaborator 11	Jr Cons./ Tech	Not completed
Collaborator 12	Jr Cons./ Tech	Not completed
Collaborator 13	Adv. Cons.	Not completed
Collaborator 14	Sr. Cons.	Not completed

As mentioned in 4.2, most experiments had a duration of thirty minutes, however, for collaborators who were available, the experimented lasted until their questionnaire was

complete. Seeing that seven of the first eleven collaborators weren't able to complete the questionnaire in thirty minutes, it was possible to conclude that it was not enough to complete the questionnaire. Either the questionnaire had to be quicker, or it should not be expected of it to be completed in such a short time and still evaluate the collaborator's knowledge correctly. This fact also let an oversight in the implementation be discovered: the system could not obtain the partial results of the assessments that were interrupted. This issue was fixed for a second round of experimentations with the last three collaborators, despite them not having completed the evaluation in the allotted time, the results were stored by the system.

During the questionnaire some collaborators stated that the questions and answers should be shorter, as reading long excerpts of text took a long time and increased the time they took to answer the questions substantially.

For the four collaborators who completed the questionnaire, the results are depicted in Table 7, in the form of three values. The first value is the starting bloom taxonomy level of the collaborator in the subtopic. The second value is the bloom taxonomy level the system concluded the user had. Lastly, the third value was the objective bloom taxonomy level that the questionnaire was testing the user for. Entries with no values mean the corresponding subtopic wasn't evaluated in that questionnaire.

Table 7 - Results of the completed questionnaires

Subtopic	Collaborator 4	Collaborator 8	Collaborator 9	Collaborator 10
Basic and structured data types	3/3/4	5/5/5	3/1/4	2/2/3
Decomposition into objects carrying state and having behavior	3/3/4	5/4/5	3/3/4	2/3/3
Class-hierarchy design for modeling	2/2/4	5/4/5	2/4/4	1/2/2
Definition of classes and fields	3/2/4	5/4/5	3/2/4	2/3/3
Definition of methods	3/0/4	5/5/5	3/3/4	2/1/3
Definition of constructors	3/1/4	5/5/5	3/1/4	1/1/3
Subclasses and Inheritance	3/2/4	5/5/5	3/2/4	1/0/3
Interfaces	3/0/4	5/4/5	3/3/4	1/0/3
Method overriding	3/1/4	5/5/5	3/2/4	1/1/3
Overloading	2/0/4	5/3/5	2/4/4	1/0/2
Dynamic dispatching	2/2/4	5/5/5	2/1/4	1/1/2

Subtopic	Collaborator 4	Collaborator 8	Collaborator 9	Collaborator 10
Exception	2/1/4	5/5/5	2/2/4	1/1/2
Subtype polymorphism; implicit upcasts in typed languages	2/0/3	5/5/5	2/3/3	0/0/2
Notion of behavioural replacement: subtypes acting like supertypes	2/0/3	5/4/5	2/3/3	0/2/2
Relationship between subtyping and inheritance	2/3/3	5/5/5	2/3/3	0/0/2
Privacy and visibility of class members	2/1/3	5/5/5	2/3/3	0/2/2
Interfaces revealing only method signatures	2/1/3	5/5/5	2/1/3	0/0/2
Using collection classes, iterators, and other common library components	1/0/3	4/4/4	1/3/3	0/1/1
Component based programming	0/0/3	4/3/4	0/0/3	-
Object delegation	0/0/3	4/4/4	0/1/3	-
Class modelling	0/1/4	5/4/5	0/2/4	-
Code refactoring	0/0/4	5/5/5	0/1/4	-
Basic Design Patterns	2/2/3	4/4/4	2/2/3	0/0/2
Advanced Design Patterns	1/0/3	4/4/4	1/3/3	0/0/1
Design principles	0/0/3	4/2/4	0/3/3	-
Object-oriented idioms for encapsulation: abstract base classes	0/1/2	4/3/4	0/2/2	-
Refactoring designs using design patterns	-	5/5/5	-	-
Architectural Patterns	-	5/2/5	-	-

Subtopic	Collaborator 4	Collaborator 8	Collaborator 9	Collaborator 10
Use of generic classes	-	3/0/3	-	-
Constrained genericity	-	3/2/3	-	-

The three collaborators that tested the system during the second round of experimentations were not able to complete the evaluations in the allotted time, however the partial results obtained were stored by the system. Table 8 presents the partial results of the evaluations done by Collaborators 12, 13 and 14 in a similar manner as the previous table but omitting the values for the subtopics whose evaluation didn't finish.

Table 8 - Second round partial results

Subtopic	Collaborator 12	Collaborator 13	Collaborator 14
Basic and structured data types	2/2/3	4/-/5	5/-/5
Decomposition into objects carrying state and having behavior	2/2/3	4/-/5	5/3/5
Class-hierarchy design for modeling	1/2/2	4/5/5	5/-/5
Definition of classes and fields	2/2/3	4/-/5	5/-/5
Definition of methods	2/-/3	4/1/5	5/5/5
Definition of constructors	1/2/3	4/-/5	5/5/5
Subclasses and Inheritance	1/-/3	4/0/5	5/5/5
Interfaces	1/0/3	4/-/5	5/5/5
Method overriding	1/-/3	4/2/5	5/-/5
Overloading	1/0/2	4/-/5	5/-/5
Dynamic dispatching	1/-/2	4/-/5	5/-/5
Exception	1/2/2	4/3/5	5/-/5
Subtype polymorphism; implicit upcasts in typed languages	0/0/2	3/-/5	5/4/5
Notion of behavioural replacement: subtypes acting like supertypes	0/1/2	3/-/5	5/-/5

Subtopic	Collaborator 12	Collaborator 13	Collaborator 14
Relationship between subtyping and inheritance	0/-/2	3/-/5	5/5/5
Privacy and visibility of class members	0/2/2	3/-/5	5/-/5
Interfaces revealing only method signatures	0/-/2	3/1/5	5/-/5
Using collection classes, iterators, and other common library components	0/-/1	3/-/4	4/-/4
Component based programming	-	3/-/4	4/-/4
Object delegation	-	3/0/4	4/-/4
Class modelling	-	4/-/5	5/-/5
Code refactoring	-	4/-/5	5/4/5
Basic Design Patterns	0/-/2	3/-/4	4/3/4
Advanced Design Patterns	0/-/1	3/-/4	4/2/4
Design principles	-	3/-/4	4/-/4
Object-oriented idioms for encapsulation: abstract base classes	-	2/-/4	4/3/4
Refactoring designs using design patterns	-	0/-/5	5/4/5
Architectural Patterns	-	0/2/5	5/5/5
Use of generic classes	-	0/-/3	3/2/3
Constrained genericity	-	0/-/3	3/2/3

According to one of the assumptions detailed in 4.2, the starting knowledge levels of the collaborators are treated as the knowledge levels demanded by their seniority level at Capgemini, and as such, are the most accurate depiction of someone's knowledge at the point in time this experimentation was conducted. The system is then considered to be evaluating a collaborator's knowledge correctly if the second value equals the first value.

From the table it is possible to observe that the system evaluated the collaborators' knowledge equal to the starting knowledge, meaning correctly, in fifty-two out of one-hundred and thirty-six evaluations, or in around thirty-eight percent (38.235%) of the subtopics.

The system evaluated the collaborator's knowledge as being lower than his starting knowledge, in fifty-five out of one-hundred and thirty-six evaluations, or in around fourty percent (40.441%) of the subtopics.

Lastly, the system concluded that the collaborator's knowledge was above the starting knowledge, in twenty-nine out of one-hundred and thirty-six evaluations, or in around twenty-one percent (21.324%) of the subtopics.

With these values in mind, and considering the assumptions made in 4.2, it is possible to conclude that the system is not fit for its intended purpose as it does not evaluate the knowledge of the collaborators correctly, only doing so in thirty-eight percent of subtopics evaluated. As such, the dynamic questionnaires algorithm must be redesigned as to create one that would more accurately evaluate a collaborator.

If one discards the assumption that the questions are fit for evaluating a collaborator's knowledge of a subtopic in a specific Bloom taxonomy level, one can place blame on the questions themselves, as they were not created by experts, unlike the ones that are going to be used in the final system.

On the other hand, if one discards the assumption that the starting knowledge level for every skill of each collaborator was the level demanded by their seniority level in the organisation, one can place blame on the assumed starting knowledge of the collaborator. As this was not inferred by the system, it might not accurately reflect the current knowledge the collaborator holds.

In summary, we will discuss the answers to the assessment objectives detailed in 4.1.

> Does the system effectively present a different questionnaire for every collaborator that used it?

Table 8 depicts the results of three incomplete questionnaires. One can see the difference in the subtopics that were evaluated in all three questionnaires. As an example, Collaborator 14 was evaluated with a Bloom taxonomy level of three ("Apply") in their knowledge of the "Basic Design Patterns" subtopic. However, Collaborator 12 and Collaborator 13 were never evaluated in that subtopic. A similar but reverse situation occurred when evaluating the "Exception" subtopic. These occurrences prove that the questionnaire was not the same for every collaborator.

> Are the questionnaires able to be completed within a specific time frame?

The specific time frame chosen was thirty minutes. From Table 6 we can see that none of the questionnaires were able to be completed in the allotted time frame.

> Are the conclusions drawn by the system similar to the expected knowledge of a collaborator?

Based on the assumptions made, the system was only able to correctly evaluate the collaborators' knowledge thirty-eight percent of the times.

5 Conclusions

This chapter presents the conclusions drawn from everything that was discussed in this document. It starts with a brief description of the contents of this document. Then it moves on to outlining the goals that were achieved and how they were achieved. Finally, suggestions are discussed for the future work to be done on the system.

In this document we started by describing the importance of human capital in organisations, and the impact of human resource management in the success of an organisation. Capgemini proposed a solution with the goal of improving their human resource management in the form of the HORUS project. The project's main goals were to better manage the development of collaborator's skills and optimise the allocation of human resources to projects within the organisation. This work focused on parts of the HORUS project, namely the modelling of a collaborator's knowledge, the determining of the engineering profile a collaborator was closest to attaining, and the evaluation of their knowledge through the use of dynamic questionnaires.

The state of the art was then presented where it was discussed the main techniques in user modelling, knowledge engines, classification algorithms, taxonomies of educational learning, and methods of measuring similarity. Similar/relevant systems to the one being developed were also presented and expounded on. Finally, it was concluded that from the research done, no system existed that fulfilled the same purpose as the one proposed, in the same context as the one the system will be applied to.

After the state of the art, the design and implementation of the system was discussed. Here we started by presenting an overview of the system, its components, and how it is going to integrate with Capgemini's Internal system. Then, every component was described in more detail, what was each one's function and how they were implemented.

Finally, the document showcased how the system was tested. First detailing the objectives of the testing process, then overviewing the two-round experimentation process through which the system was tested, and lastly presenting the results obtained in the experimentations and discussing them.

This system's development is not concluded and will be further iterated on by Capgemini Engineering and the HORUS project team.

5.1 Goals Achieved

To better understand what goals were achieved by the proposed system, we will provide answers to the research questions proposed in 1.3. Then we will detail which functional requirements were met, and which use cases were implemented.

1. Can we model a collaborator's knowledge and the knowledge required to attain an engineering profile in a comparable form?

It is not possible to answer this question conclusively, as no testing was conducted regarding it. We separated the domain knowledge into subtopics, whose collaborators' knowledge of were classified according to the first five levels of the Bloom taxonomy ("Remember", "Understand", "Apply", "Analyse" and "Evaluate"). Subtopics can then be incorporated into several skills, classified on a scale of four knowledge levels ("Beginner", "Intermediate", "Advanced" and "Expert"). Since both the collaborators' knowledge and the knowledge required by engineering profiles are composed of these same pieces, they can then be compared. A way to test if this approach is adequate will be suggested in 5.2.

2. Can we identify an engineering profile a collaborator is close to attaining?

Similarly, we cannot answer this question, as no testing was conducted regarding it. Since the collaborators and the engineering profiles could be compared, the only step remaining was to measure the similarity between one another. This measurement was made by transforming the knowledge into n-dimensional vector representations of the collaborators and engineering profiles, and then measuring the distance between them using the Euclidean distance. The engineering profile with the least distance to the collaborator was then set as a goal for the collaborator to achieve. However, this does not mean that providing a concrete goal for the collaborator results in an increase in productivity or morale. A way to test if this approach is adequate will be suggested in 5.2.

3. Can we accurately evaluate an employee's knowledge through the use of dynamic questionnaires?

The answer as of now is no. The implemented dynamic questionnaires algorithm only correctly evaluated the knowledge of a collaborator thirty-eight percent of the times. As such it proved itself not to be a viable option in accurately evaluating collaborators' knowledge of the domain. However, the author feels that some changes to the algorithm can increase its performance into an acceptable level. This belief is furthered by the existence of the similar system that uses dynamic questionnaires presented in 2.6.

Table 9 presents the status of the functional requirements defined in 3.2.

Table 9 - Functional Requirement status

Functional Requirement	Status
R01	Implemented, though results are dubious
R02	Implemented
R03	Implemented, not tested

All functional requirements were implemented, although requirement R03 was not tested. R01's results of the evaluations are not accurate and cannot be considered the knowledge held by a collaborator.

Table 10 presents the status of the use cases defined in 3.2.

Table 10 - Use Case status

Use Case	Status
UC01	Implemented
UC02	Implemented

Both defined use cases were implemented.

Considering the answers to the research questions, and the status of the functional requirements and use cases, one can conclude that there is still much room for improvement. Namely testing the system implemented and making changes to the evaluation algorithm, as to obtain better results.

5.2 Future Work

As the system is not even nearing completion, there is much potential to improve it in the future. These can either be improvements to the current functionalities, or the implementation of secondary functionalities that could be used to ease in the use of the system. Following are some suggestions:

- Storing in the database every answer the collaborator gave to questions in the questionnaire, instead of just the collaborator's knowledge inferred by the system;
- Implement a functionality that could import questions from the future crowdsourcing platform into the database, and another that could transform the questions in the database directly into a ".kqb" file for Pyke to use;
- Make the system fetch the results of new users' self-evaluation to serve as their starting knowledge, when access is provided to the self-evaluation functionalities of Capgemini's internal system;
- Test the system with expert-created questions instead of the ones created by the HORUS project team;

- Add a functionality for the system to suggest study materials for collaborators to use according to the knowledge they lack;
- Test the first research question by consulting experts in the field of knowledge engineering, and making the necessary changes based on their feedback;
- Test the second research question by measuring the length of time it takes for collaborators to attain the nearest (calculated by the system) engineering profile. And compare it to the length of time it takes for a collaborator to attain an engineering profile not suggested by the system.

The development of the presented system will be continued by Capgemini Engineering, using it as a base to build upon. Upon further testing and results obtained from the use of this system in a real environment, the need may arise for other functionalities to be implemented or changes to be made to existing ones.

5.3 Author Remarks

I believe the implemented system has contributed significantly to the HORUS project, as its team will be able to build on the foundations set, and further improve the system. I think there is much potential for this system to be used in the continuous development of collaborators' skills and optimisation of the allocation of human resources to projects within an organisation.

I also believe that despite the negative result during the system assessment, it could have performed vastly better without the assumptions made at the start of the experimentation process. Meaning, had we access to the self-evaluation of each collaborator that tested the system instead of having to infer the collaborator's starting knowledge, and had we used questions created by experts instead of ones created by the HORUS project's team, the system would have fared far better.

References

[1] M. Vokoun, Z. Caha, J. Straková, and F. Stellner, 'THE STRATEGIC IMPORTANCE OF HUMAN RESOURCES MANAGEMENT AND THE ROLES OF HUMAN CAPITAL INVESTMENT AND EDUCATION', p. 11.

[2] V. Terziev, 'IMPORTANCE OF HUMAN RESOURCES TO SOCIAL DEVELOPMENT', no. 12, p. 9, 2018.

[3] 'cept-website-doc-horus-v2.pdf'. https://capgemini-engineering.com/as-content/uploads/sites/27/2022/02/cept-website-doc-horus-v2.pdf (accessed Oct. 05, 2022).

[4] R. D. Mendes, 'INTELIGÊNCIA ARTIFICIAL: SISTEMAS ESPECIALISTAS NO GERENCIAMENTO DA INFORMAÇÃO', *Ciênc. Informação*, vol. 26, pp. 39–45, Jan. 1997, doi: 10.1590/S0100-19651997000100006.

[5] J. M. S. da Rocha, 'Concepção e implementação de um sistema pericial no domínio da cardiologia', 1990.

[6] J. A. Matelli, E. Bazzo, and J. C. da Silva, 'An expert system prototype for designing natural gas cogeneration plants', *Expert Syst. Appl.*, vol. 36, no. 4, pp. 8375–8384, May 2009, doi: 10.1016/j.eswa.2008.10.083.

[7] J. Clancey, 'F/G 9/2 THE EPISTEMOLOGY OF A RULE-BASED EXPERT SYSTEM: A FRAMEWORK FOR--ETC(Ul', p. 79.

[8] D. Janjanam, B. Ganesh, and L. Manjunatha, 'Design of an expert system architecture: An overview', *J. Phys. Conf. Ser.*, vol. 1767, no. 1, p. 012036, Feb. 2021, doi: 10.1088/1742-6596/1767/1/012036.

[9] L. D. Otero and C. E. Otero, 'A fuzzy expert system architecture for capability assessments in skill-based environments', *Expert Syst. Appl.*, vol. 39, no. 1, pp. 654–662, Jan. 2012, doi: 10.1016/j.eswa.2011.07.057.

[10] C. Cucu, 'EDUCATIONAL OBJECTIVES IN INTELLIGENT TUTORS FOR COMMUNICATION SKILLS', p. 8.

[11] F. Grivokostopoulou, I. Perikos, and I. Hatzilygeroudis, 'An intelligent tutoring system for teaching FOL equivalence', in *The First Workshop on AI-supported Education for Computer Science (AIEDCS 2013)*, 2013, vol. 20.

[12] F. Grivokostopoulou, I. Perikos, and I. Hatzilygeroudis, 'An Educational System for Learning Search Algorithms and Automatically Assessing Student Performance', *Int. J. Artif. Intell. Educ.*, vol. 27, no. 1, pp. 207–240, Mar. 2017, doi: 10.1007/s40593-016-0116-x.

[13] ‘Applied Sciences | Free Full-Text | Design of an Intelligent Tutoring System to Create a Personalized Study Plan Using Expert Systems | HTML’. https://www.mdpi.com/2076-3417/12/12/6236/htm (accessed Oct. 17, 2022).

[14] F. Montalvo-Ochoa, V. Robles-Bykbaev, P. Duque-Sarango, and K. González-Arias, ‘An educational rule-based expert system to determine water quality for environmental engineering and biotechnology students’, in *2020 IEEE World Conference on Engineering Education (EDUNINE)*, Mar. 2020, pp. 1–6. doi: 10.1109/EDUNINE48860.2020.9149502.

[15] C. P. Lee, Z. B. Ng, Y. E. Low, and K. M. Lim, ‘Expert System for University Program Recommendation’, in *2020 IEEE 2nd International Conference on Artificial Intelligence in Engineering and Technology (IICAIET)*, Sep. 2020, pp. 1–6. doi: 10.1109/IICAIET49801.2020.9257822.

[16] ‘Drools - Business Rules Management System (Java™, Open Source)’, *Drools*. https://drools.org/ (accessed Oct. 09, 2022).

[17] ‘PyCLIPS Python Module | Python and CLIPS Integration’. https://pyclips.sourceforge.net/web/ (accessed Oct. 09, 2022).

[18] ‘SWI-Prolog’s features’. https://www.swi-prolog.org/features.html (accessed Oct. 09, 2022).

[19] ‘Welcome to Pyke’. https://pyke.sourceforge.net/index.html (accessed Oct. 09, 2022).

[20] P. A. Jaques *et al.*, ‘Rule-based expert systems to support step-by-step guidance in algebraic problem solving: The case of the tutor PAT2Math’, *Expert Syst. Appl.*, vol. 40, no. 14, pp. 5456–5465, Oct. 2013, doi: 10.1016/j.eswa.2013.04.004.

[21] R. A. S. Jr, ‘Partial Automation of the Cartographic Design Process’, p. 255.

[22] J. Singla, ‘The Diagnosis of Some Lung Diseases in a Prolog Expert System’, *Int. J. Comput. Appl.*, vol. 78, no. 15, pp. 37–40, Sep. 2013, doi: 10.5120/13603-1435.

[23] X. Zhang, G. P. Moynihan, A. N. S. Ernest, and J. L. Gutenson, ‘Evaluation of the benefits of using a backward chaining decision support expert system for local flood forecasting and warning’, *Expert Syst.*, vol. 35, no. 4, p. e12261, 2018, doi: 10.1111/exsy.12261.

[24] M. Elsom-Cook, ‘Student modelling in intelligent tutoring systems’, *Artif. Intell. Rev.*, vol. 7, no. 3–4, pp. 227–240, Aug. 1993, doi: 10.1007/BF00849556.

[25] G. Fischer, ‘User Modeling in Human–Computer Interaction’, *User Model. User-Adapt. Interact.*, vol. 11, no. 1, pp. 65–86, Mar. 2001, doi: 10.1023/A:1011145532042.

[26] S. Abri, R. Abri, and S. Cetin, ‘A Classification on Different Aspects of User Modelling in Personalized Web Search’, in *Proceedings of the 4th International Conference on Natural*

Language Processing and Information Retrieval, Seoul Republic of Korea, Dec. 2020, pp. 194–199. doi: 10.1145/3443279.3443291.

[27] S. Abri, R. Abri, and S. Çetin, 'Group-based Personalization Using Topical User Profile', in *Adjunct Publication of the 28th ACM Conference on User Modeling, Adaptation and Personalization*, New York, NY, USA, Jul. 2020, pp. 181–186. doi: 10.1145/3386392.3399559.

[28] Y. Zhu, L. He, and X. Wang, 'User Interest Modeling and Self-Adaptive Update Using Relevance Feedback Technology', *Procedia Eng.*, vol. 29, pp. 721–725, Jan. 2012, doi: 10.1016/j.proeng.2012.01.030.

[29] K. Sugiyama, K. Hatano, and M. Yoshikawa, 'Adaptive Web search based on user profile constructed without any effort from users', presented at the Thirteenth International World Wide Web Conference Proceedings, WWW2004, Jan. 2004, pp. 675–684. doi: 10.1145/988672.988764.

[30] D. Poo, B. Chng, and J.-M. Goh, 'A hybrid approach for user profiling', in *36th Annual Hawaii International Conference on System Sciences, 2003. Proceedings of the*, Jan. 2003, p. 9 pp.-. doi: 10.1109/HICSS.2003.1174242.

[31] J. Fink and A. Kobsa, 'A Review and Analysis of Commercial User Modeling Servers for Personalization on the World Wide Web', *User Model. User-Adapt. Interact.*, vol. 10, no. 2, pp. 209–249, Jun. 2000, doi: 10.1023/A:1026597308943.

[32] J. E. Greer and G. I. McCalla, *Student Modelling: The Key to Individualized Knowledge-Based Instruction*. Springer Science & Business Media, 2013.

[33] M. J. Mayo, 'Bayesian Student Modelling and Decision-Theoretic Selection of Tutorial Actions in Intelligent Tutoring Systems', 2001, doi: 10.26021/1918.

[34] B. P. Woolf, 'Student Modeling', in *Advances in Intelligent Tutoring Systems*, R. Nkambou, J. Bourdeau, and R. Mizoguchi, Eds. Berlin, Heidelberg: Springer, 2010, pp. 267–279. doi: 10.1007/978-3-642-14363-2_13.

[35] S. Sani, T. Mohd Aris, and M. Sulaiman, 'Student Modeling: An Overview', *Int. J. Adv. Comput. Sci. Its Appl.*, vol. 5, pp. 2250–3765, Oct. 2015.

[36] K. Chrysafiadi and M. Virvou, 'Student modeling approaches: A literature review for the last decade', *Expert Syst. Appl.*, vol. 40, no. 11, pp. 4715–4729, Sep. 2013, doi: 10.1016/j.eswa.2013.02.007.

[37] A. C. Martins and E. Carrapatoso, 'USER MODELING IN ADAPTIVE HYPERMEDIA EDUCATIONAL SYSTEMS', p. 14.

[38] R. R. Burton and J. S. Brown, 'A tutoring and student modelling paradigm for gaming environments', *ACM SIGCSE Bull.*, vol. 8, no. 1, pp. 236–246, Feb. 1976, doi: 10.1145/952989.803477.

[39] A. Mitrovic, 'Modeling Domains and Students with Constraint-Based Modeling', in *Advances in Intelligent Tutoring Systems*, R. Nkambou, J. Bourdeau, and R. Mizoguchi, Eds. Berlin, Heidelberg: Springer, 2010, pp. 63–80. doi: 10.1007/978-3-642-14363-2_4.

[40] B. Martin, 'Constraint-based modelling: Representing student knowledge', *N. Z. J. Comput.*, vol. 7, no. 2, pp. 30–38, 1999.

[41] N. Khodeir, N. Wanas, and H. Elazhary, 'Constraint-based Student Modelling in Probability Story Problems with Scaffolding Techniques', *Int. J. Emerg. Technol. Learn. IJET*, vol. 13, p. 178, Jan. 2018, doi: 10.3991/ijet.v13i01.7397.

[42] J. Kay, 'Stereotypes, Student Models and Scrutability', in *Intelligent Tutoring Systems*, Berlin, Heidelberg, 2000, pp. 19–30. doi: 10.1007/3-540-45108-0_5.

[43] X. Zhang and H. Han, 'An empirical testing of user stereotypes of information retrieval systems', *Inf. Process. Manag.*, vol. 41, no. 3, pp. 651–664, May 2005, doi: 10.1016/j.ipm.2004.01.005.

[44] C. Zhang, C. Liu, X. Zhang, and G. Almpanidis, 'An up-to-date comparison of state-of-the-art classification algorithms', *Expert Syst. Appl.*, vol. 82, pp. 128–150, Oct. 2017, doi: 10.1016/j.eswa.2017.04.003.

[45] T. A. Cardona and E. a. Cudney, 'Predicting Student Retention Using Support Vector Machines', *Procedia Manuf.*, vol. 39, pp. 1827–1833, Jan. 2019, doi: 10.1016/j.promfg.2020.01.256.

[46] S. Wiyono and T. Abidin, 'COMPARATIVE STUDY OF MACHINE LEARNING KNN, SVM, AND DECISION TREE ALGORITHM TO PREDICT STUDENT'S PERFORMANCE', *Int. J. Res. - GRANTHAALAYAH*, vol. 7, no. 1, Art. no. 1, Jan. 2019, doi: 10.29121/granthaalayah.v7.i1.2019.1048.

[47] A. Slim, D. Hush, T. Ojah, and T. Babbitt, *Predicting Student Enrollment Based on Student and College Characteristics*. International Educational Data Mining Society, 2018. Accessed: Jan. 28, 2022. [Online]. Available: https://eric.ed.gov/?id=ED593221

[48] M. Aghaabbasi, Z. A. Shekari, M. Z. Shah, O. Olakunle, D. J. Armaghani, and M. Moeinaddini, 'Predicting the use frequency of ride-sourcing by off-campus university students through random forest and Bayesian network techniques', *Transp. Res. Part Policy Pract.*, vol. 136, pp. 262–281, Jun. 2020, doi: 10.1016/j.tra.2020.04.013.

[49] R. Hasan, S. Palaniappan, S. Mahmood, A. Abbas, K. U. Sarker, and M. U. Sattar, 'Predicting Student Performance in Higher Educational Institutions Using Video Learning Analytics and Data Mining Techniques', *Appl. Sci.*, vol. 10, no. 11, Art. no. 11, Jan. 2020, doi: 10.3390/app10113894.

[50] D. Hao and D. Chai, 'Application of SVM-KNN intelligent classification prediction model in IT Vocational Education', in *2021 5th Asian Conference on Artificial Intelligence Technology (ACAIT)*, Oct. 2021, pp. 312–315. doi: 10.1109/ACAIT53529.2021.9731162.

[51] B. Pérez, C. Castellanos, and D. Correal, 'Predicting Student Drop-Out Rates Using Data Mining Techniques: A Case Study', in *Applications of Computational Intelligence*, Cham, 2018, pp. 111–125. doi: 10.1007/978-3-030-03023-0_10.

[52] S. Vanaja and K. Rameshkumar, 'Performance analysis of classification algorithms on medical diagnoses-a survey', *J. Comput. Sci.*, vol. 11, no. 1, p. 31, 2015.

[53] D. R. Cox, 'The Regression Analysis of Binary Sequences', *J. R. Stat. Soc. Ser. B Methodol.*, vol. 20, no. 2, pp. 215–242, 1958.

[54] S. U. Portl, 'Logistic Regression', *Categ. Data Anal.*, p. 13, 2021.

[55] J. Joyce, 'Bayes' Theorem', Jun. 2003, Accessed: Jan. 23, 2022. [Online]. Available: https://plato.stanford.edu/archives/spr2019/entries/bayes-theorem/

[56] C. Cortes and V. Vapnik, 'Support-vector networks', *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995, doi: 10.1007/BF00994018.

[57] T. O. Ayodele, 'Types of machine learning algorithms', *New Adv. Mach. Learn.*, vol. 3, pp. 19–48, 2010.

[58] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, 'Supervised machine learning: A review of classification techniques', *Emerg. Artif. Intell. Appl. Comput. Eng.*, vol. 160, no. 1, pp. 3–24, 2007.

[59] N. S. Altman, 'An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression', *Am. Stat.*, vol. 46, no. 3, pp. 175–185, Aug. 1992, doi: 10.1080/00031305.1992.10475879.

[60] J E T Akinsola, 'Supervised Machine Learning Algorithms: Classification and Comparison', *Int. J. Comput. Trends Technol. IJCTT*, vol. 48, pp. 128–138, Jun. 2017, doi: 10.14445/22312803/IJCTT-V48P126.

[61] L. Breiman, 'Random Forests', *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001, doi: 10.1023/A:1010933404324.

[62] S. Lessmann, B. Baesens, H.-V. Seow, and L. C. Thomas, 'Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research', *Eur. J. Oper. Res.*, vol. 247, no. 1, pp. 124–136, Nov. 2015, doi: 10.1016/j.ejor.2015.05.030.

[63] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, 'Do we Need Hundreds of Classifiers to Solve Real World Classification Problems?', *J. Mach. Learn. Res.*, vol. 15, no. 90, pp. 3133–3181, 2014.

[64] J. Gama and P. Brazdil, *Characterization of Classification Algorithms.* 1995, p. 200. doi: 10.1007/3-540-60428-6_16.

[65] J. Tabak, *Geometry: The Language of Space and Form*. Infobase Publishing, 2014.

[66] W. Supanich and S. Kulkarineetham, 'Personalized Tourist Attraction Recommendation System Using Collaborative Filtering on Tourist Preferences', in *2022 19th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, Jun. 2022, pp. 1–6. doi: 10.1109/JCSSE54890.2022.9836255.

[67] A. Kumar, S. F. Khan, R. S. Sodhi, I. R. Khan, S. Kumar, and A. K. Tamrakar, 'Deep learning Based Patient-Friendly Clinical Expert Recommendation Framework', in *2022 2nd International Conference on Innovative Practices in Technology and Management (ICIPTM)*, Feb. 2022, vol. 2, pp. 736–741. doi: 10.1109/ICIPTM54933.2022.9754157.

[68] A. Singhal, 'Modern Information Retrieval: A Brief Overview', p. 9.

[69] A. D. Hartanto, Y. Pristyanto, and A. Saputra, 'Document Similarity Detection using Rabin-Karp and Cosine Similarity Algorithms', in *2021 International Conference on Computer Science and Engineering (IC2SE)*, Nov. 2021, vol. 1, pp. 1–6. doi: 10.1109/IC2SE52832.2021.9791999.

[70] Y. Januzaj and A. Luma, 'Cosine Similarity – A Computing Approach to Match Similarity Between Higher Education Programs and Job Market Demands Based on Maximum Number of Common Words', *Int. J. Emerg. Technol. Learn. IJET*, vol. 17, no. 12, Art. no. 12, Jun. 2022, doi: 10.3991/ijet.v17i12.30375.

[71] K. Yeager, 'LibGuides: SPSS Tutorials: Pearson Correlation'. https://libguides.library.kent.edu/SPSS/PearsonCorr (accessed Oct. 17, 2022).

[72] P. Devika, K. Jyothisree, P. Rahul, S. Arjun, and J. Narayanan, 'Book Recommendation System', in *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, Jul. 2021, pp. 1–5. doi: 10.1109/ICCCNT51525.2021.9579647.

[73] S. Sadia, M. B. Propa, K. S. Al Mamun, and M. S. Kaiser, 'A Fruit Cultivation Recommendation System based on Pearson's Correlation Co-Efficient', in *2021 International Conference on Information and Communication Technology for Sustainable Development (ICICT4SD)*, Feb. 2021, pp. 361–365. doi: 10.1109/ICICT4SD50815.2021.9396923.

[74] B. S. Bloom, M. D. Engelhart, E. J. Furst, W. H. Hill, and D. R. Krathwohl, 'Handbook I: cognitive domain', *N. Y. David McKay*, 1956.

[75] D. L. D. Fink, 'WHAT IS '"SIGNIFICANT LEARNING"?', p. 8.

[76] E. J. Simpson, *THE CLASSIFICATION OF EDUCATIONAL OBJECTIVES, PSYCHOMOTOR DOMAIN*. 1966. Accessed: Oct. 08, 2022. [Online]. Available: https://eric.ed.gov/?id=ED010368

[77] J. Conklin, ‘Review of A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom’s Taxonomy of Educational Objectives Complete Edition’, *Educ. Horiz.*, vol. 83, no. 3, pp. 154–159, 2005.

[78] ‘Bloom’s Taxonomy’, *Vanderbilt University*. https://cft.vanderbilt.edu/guides-sub-pages/blooms-taxonomy/ (accessed Oct. 08, 2022).

[79] M. Solem, K. Foote, and J. Monk, Eds., *Aspiring academics: a resource book for graduate students and early career faculty*. Upper Saddle River, NJ: Pearson/Prentice Hall, 2009.

[80] P. Molins-Ruano, C. González-Sacristán, F. Díez, P. R. Marín, and S. Gomez-Monivas, ‘Adaptive Model for Computer-Assisted Assessment in Programming Skills’, *ArXiv*, 2014.

[81] P. Molins-Ruano, F. Borrego Gallardo, C. Sevilla, F. Jurado, P. Rodríguez, and G. M. Sacha, ‘Construcción de cuestionarios de calidad con e-valUAM.’, *Proc. XVI Simp. Int. Informática Educ. Acceso Masivo Univers. Para Un Aprendizajea Lo Largo Vida*, pp. 291–298, Jan. 2014.

[82] P. Molins-Ruano, P. Rodriguez, S. Atrio, and G. M. Sacha, ‘Modelling experts’ behavior with e-valUAM to measure computer science skills’, *Comput. Hum. Behav.*, vol. 61, pp. 378–385, Aug. 2016, doi: 10.1016/j.chb.2016.03.044.

[83] E. Gaudioso, M. Montero, and F. Hernandez-del-Olmo, ‘Supporting teachers in adaptive educational systems through predictive models: A proof of concept’, *Expert Syst. Appl.*, vol. 39, no. 1, pp. 621–625, Jan. 2012, doi: 10.1016/j.eswa.2011.07.052.

[84] S. Durrani and D. S. Durrani, ‘Intelligent tutoring systems and cognitive abilities’, in *Proceedings of graduate colloquium on computer sciences (GCCS)*, 2010.

[85] G.-M. Baschera and M. Gross, ‘Poisson-Based Inference for Perturbation Models in Adaptive Spelling Training’, *Int. J. Artif. Intell. Educ.*, vol. 20, no. 4, pp. 333–360, Jan. 2010, doi: 10.3233/JAI-2010-011.

[86] C. Bonesana, F. Mangili, and A. Antonucci, ‘ADAPQUEST: A Software for Web-Based Adaptive Questionnaires based on Bayesian Networks’. arXiv, Dec. 29, 2021. Accessed: Oct. 19, 2022. [Online]. Available: http://arxiv.org/abs/2112.14476

[87] ‘MySQL :: MySQL Connector/Python Developer Guide’. https://dev.mysql.com/doc/connector-python/en/ (accessed Oct. 10, 2022).

[88] ‘scikit-learn: machine learning in Python — scikit-learn 1.1.2 documentation’. https://scikit-learn.org/stable/index.html (accessed Oct. 10, 2022).

[89] ‘Rules’. https://pyke.sourceforge.net/logic_programming/rules/index.html (accessed Sep. 30, 2022).