2022 3rd International Conference on Education, Knowledge and Information Management (ICEKIM) | 978-1-6654-1666-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICEKIM55072.2022.00076

Design of Domain-driven Microservices-based Software Talent Evaluation and Recommendation System

Kun ZHANG
Huangshan University
Huangshan, China
e-mail: hszk@hsu.edu.cn

Ji TIAN
Huangshan University
Huangshan, China

Qin Yuan
Huangshan University
Huangshan, China

*Jing Han
Huangshan University
Huangshan, China
* Corresponding author: 105039@hsu.edu.cn

***Abstract*—Software talent ability evaluation exists both on the school side and the market side. Companies cannot fully understand the ability of candidates through resumes and interviews in a short period. In response to this situation, this paper proposes a microservices-based evaluation and recommendation system that attempts to bridge the supply and demand sides of talent. We collect multi-dimensional data on students' learning process and engineering project activities during their school years from the school side, use the crawled job requirement data to build the occupational competency requirement model from the market side, and finally use similarity calculation to achieve talent recommendation.**

Keywords-domain-driven design; microservices; competency requirement model; knowledge graph; recommendation algorithm

I. INTRODUCTION

Unequal information of students between universities and companies often leads to the problem of mismatch which affects the recruitment of companies and the reputation of schools. Universities can collect almost all information about students in school, but often the data given to companies is based on course grades, and the data is biased not able to fully demonstrate the multidimensional capabilities of students; Companies have the direct market demand, but they cannot get a relatively complete impression of a student through the resume, tests and interviews in a short period. Besides these problems, the authenticity and reliability of data obtained by the companies should also be concerned.

In order to get through the information barrier between supply side and demand side, it is necessary to design a more comprehensive and scientific collection system for data of student and market demand, and use talent recommendation algorithms to form automatic recommendations for companies. The software talent evaluation and recommendation system (STERS) take software engineering students and related market enterprises as the service targets, attempts to improve the comprehensiveness, authenticity and reliability of collected data through user behavior data collection and algorithm model validation. The system design can provide basic student data for teaching and research, provide more fair and comprehensive student evaluation information, shorten the distance between universities and the market, and can be used as an effective example of the construction of educational informatization in universities.

II. DATA COLLECTION ANALYSIS

A. Capability classification and collection methods

1) Capability classification

According to competency framework for software engineers [1], combined with the team's teaching reform research experience and the results of job market and training market research, qualified software engineering students should have the necessary professional skills, engineering project capabilities and other necessary excellent work qualities on the basis of solid mathematical and English skills. The specific content is divided as follows:

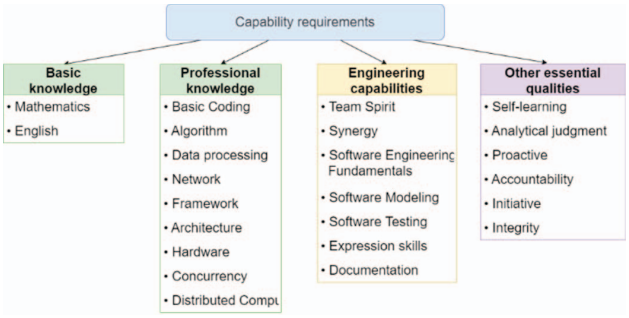

Fig 1 Software talent capability requirements table

2) Capability calculability analysis and data collection method

In book *computer and intelligence* [2], Turing discussed the possibility of simulating cognitive process with computable mathematical functions, that is, the computability of cognition. Talent assessment through machine computing is essentially a cognitive process of capability through mathematical functions. The optimal state for this system is to achieve the entire cognition process by functions, however, considering that some of the evaluations in competency literacy are subjective (e.g., initiative, integrity, etc.), and could not be improved effectively by existing AI algorithms, we introduced subjective evaluation

978-1-6654-1666-5/22/$31.00 ©2022 IEEE
DOI 10.1109/ICEKIM55072.2022.00076

(e.g., teacher evaluation, peer evaluation, etc.) to fill this part so a relatively complete evaluation result can be obtained.

According to the above analysis and the classifications in figure 1, software engineering capabilities can be divided into three main parts: knowledge based capabilities, engineering capabilities and other essential qualities. The computability of each component is discussed below:

- *Knowledge based capability data collection*

Knowledge based capabilities are mainly related to the mastery level of basic courses (Mathematics, English…) and specialized courses (Data Structures, Computer Networks…). The homework scores and examination scores of the course are the major part of the collecting data. An online learning and testing system with a knowledge database (knowledge graphs and relational algorithms) could be set up to evaluate students' knowledge mastery level, it could also record students' online using process data. This type of verification system has been effectively applied to various online education platforms, which can better realize the relevant ability calculation.

- *Engineering capabilities data collection*

Engineering capability is the overall evaluation of the cooperation ability, task scheduling ability, personnel resource coordination and other abilities for talents in a team. Some of these data, such as project task completion rate, meeting frequency, and advancement, can be collected through the task management system, but the determination of teamwork, personnel coordination, and other capabilities is more subjective, so we use the teacher's evaluation as the main method and peer evaluation as the supplement to measure this part of the data.

- *Other essential qualities data collection*

Other essential qualities evaluations such as self-learning ability and analytical judgment ability are more subjective and difficult to measure by automatic data collection, so we used the same way as the engineering capability assessment, using the teacher's opinion as the main method, and the team members' mutual evaluation data as the auxiliary form, using the weighted calculation to complete the assessment of the relevant competency, and finally to achieve the normalization calculation of this data.

3) Authenticity Guarantee

The normal data in the system is divided into automatically measured parts (such as knowledge learning and testing) and subjective evaluation parts. To improve the authenticity of the normal data, the system also records and analyzes the user's usage behavior (login, filling, copying data, time information, etc.).

For the part of automatic measurement, the system adds the identification of common cheating behaviors, such as detection of "too short" homework time, repetition detection, etc., and reduces the score of related data and the integrity score in the subjective evaluation; For the subjective evaluation part, teachers are only checked and reminded in the case of large disparity between their evaluation and team member evaluation scores.

4) Reliability Guarantee

STERS uses microservices to improve system reliability. Microservices structure makes the services distributed, and usually the offline of a single service does not bring down the overall service. At the same time, the system uses service registration middleware to monitor the service status, ensuring that the relevant managers can be alerted in time when problems arise.

B. Market requirement data collection

The market demand mainly comes from job websites, such as 51job, boss, and indeed. In the case of legal compliance, the system uses crawlers to get the job information of software companies by keyword (IT, software, etc.). The collected information includes job classification (front end, full stack), technical stack (Java, Python), job description, job requirements, experience requirements, etc., forming a job requirement information database.

III. KEY TECHNOLOGIES

A. Microservices structure

Microservices architecture is a style of architecture that started in 2014[3]. It advocates dividing a single application into a set of small services, each running independently in its own process, and coordinating with each other. The services communicate with each other using lightweight communication mechanisms, such as HTTP RESTful APIs. Each service is built around a specific business and can be deployed independently to production environment.

Before microservices, monolithic development pattern was mainly used, and systems usually adopted a layered model, such as the MVC architecture. However, this type of development model requires a specific technology stack for each project, which has limitations in deployment and expansion with high coupling between services, and has high operation and maintenance costs.

Considering that STERS contains multiple sub platforms such as knowledge learning and testing, project task management, and may continue to expand due to the increase of data sources, using a monolithic development model may not be a good choice which may lead to the duplication of work and is not conducive to data interaction between platforms, therefore, the system use microservices structure, and in order to prevent the complexity of managing the large number of microservices in the development, a domain-driven model was adopted in the design process.

B. Domain-driven design (DDD)

Eric Evans proposed the concept of DDD[4], deconstructing the traditional software engineering architecture design problems from the business perspective. Through boundary and context division, DDD breaks down and simplifies complex business logic, which is more suitable for the current status of social service system.

Monolithic (traditional) development is more targeted, the system could be divided into dozens of isolated website platforms, but it also brings higher difficulty and duplication of work for data integration, service expansion, and version

extension (Android, IOS, etc.). Microservice architecture design splits the overall requirements, which can better improve the development efficiency and reduce the development difficulty, however, hundreds of services bring huge complexity to management. The domain division design approach can clearly distinguish the business logic while evaluating the system as a whole, which is an important guide for the later detailed design.

After an in-depth study and discussion of the business processes of STERS, the main domain part of the system is divided and designed as follows:

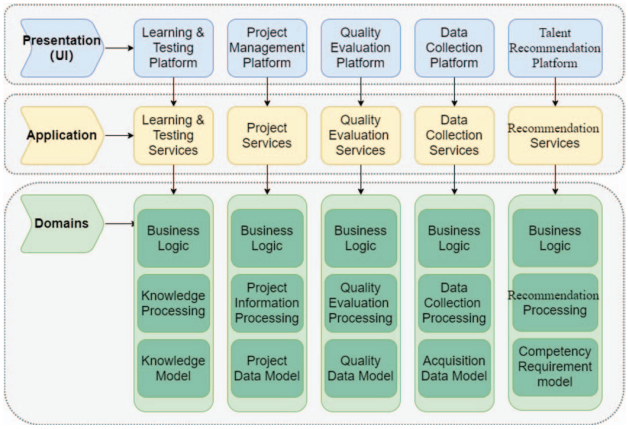

Fig 2 Main domain division chart

Taking individual capabilities classification as the general domain division guideline, we use the representation layer to implement various types of user interaction behaviors, the application layer to implement the service management and the domain layer to implement the business logic.

Knowledge learning and testing, project management and scoring platform constitute the evaluation system of different capabilities.

Data collection platform is divided into two parts, one part collects the usage data of various users in the system, and the other part crawls the required job demand data from internet.

Calculation and talent recommendation platform uses the data model generated from multiple data sources in the system to perform similarity calculation to realize the talent recommendation.

According to the domain division principle, the domain can be divided into sub-domains, and the contextual boundaries between the sub-domains are delineated by domain boundary analysis. Taking the above knowledge processing domain as an example, it can be subdivided into the following domains:

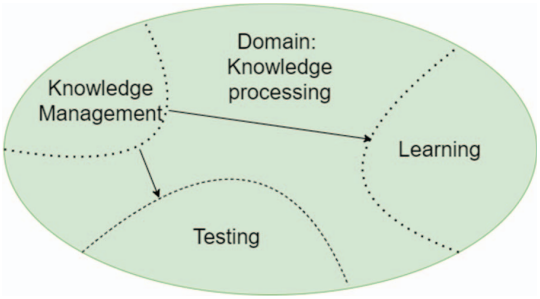

Fig 3 Knowledge processing sub-domain segmentation

Based on the above operations, a refined set of sub domains can be formed, providing clear boundaries for the segmentation and management of microservices.

C. *Knowledge graph and related computing*

Coming from philosophy, Ontology attempts to formalize the specification of shared conceptual models[5]. Semantic Web (Web) technologies adds metadata to World Wide Web documents using RDF triples, attempt to transform the Internet into a large knowledge network[6].

The above researches have promoted the development of knowledge graph technology, bringing theoretical and engineering basis for knowledge extraction, storage, utilization. At present, knowledge graphs are mainly stored in the form of RDF triples (XML databases) and graph databases.

As an important part of STERS, the knowledge-based capability data collection needs to use an effective way to store and manage the knowledge structure, using it as the underlying support for the learning (learning recommendation) and testing (automatic test generation) system.

STERS uses the empirical model to construct the knowledge graph of online courses, storing and querying the knowledge structure with the help of Neo4j graph database. The knowledge graph meta-path algorithm is used to form the recommendation of the learning content. The following diagram shows part of the knowledge graph of the C language:

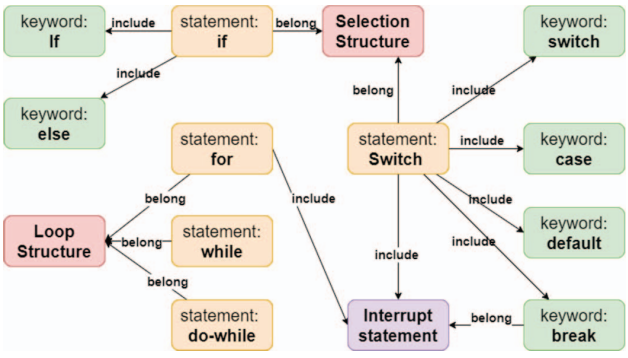

Fig 4 Knowledge graph of C language (part)

D. *Kmeans-based competency requirement model and talent recommendation algorithm*

The Kmeans algorithm is an important unsupervised clustering classification algorithm that centers on K samples

given or automatically selected, iteratively aggregates and classifies data from a learning dataset by calculating vector's similarity[7]. In SERTS, we choose the sum of squares due to error as the object function. The formula is:

$$SSE = \sum_{i=1}^{k} \sum_{x \in C_i} (C_i - x)^2 \quad (1)$$

SERTS crawls job information once a week. After the data cleaning step, a matrix will be constructed by different requirements (job requirements, technical stack, skills requirements, engineering ability requirements, education requirements, etc.), using Euclidean distance, the collected job information is clustered using KMeans algorithm to form a competency requirement model. The number K in this process is calculated using the following equation:

$$K \approx \sqrt{n/2} \quad (2)$$

since the difference in technology stacks has a large impact on the recruitment results, the technology stack data was weighted in the normalization process.

A feature model around K particles can be formed after the algorithm above, and then we can calculate the Euclidean distance between the talent data and the model, and thus complete the ranking and recommendation for talents.

IV. System Design and Implementation

A. Platforms organization

The school side of SERTS is a multi-platform structure formed by a large number of microservices, including independent functional platforms (knowledge learning and testing, project management, qualities evaluation) and cross-platform service clusters (internal data collection, user authentication, cloud disk services, etc.); For the market side, SERTS cleans the crawled data to form the demand database and uses Kmeans algorithm to form the market demand model. This system builds a computing layer between the two sides, extract data from the school side and compare it with the market demand model to realize the automatic recommendation of talents. The following figure shows the relationships between platforms:

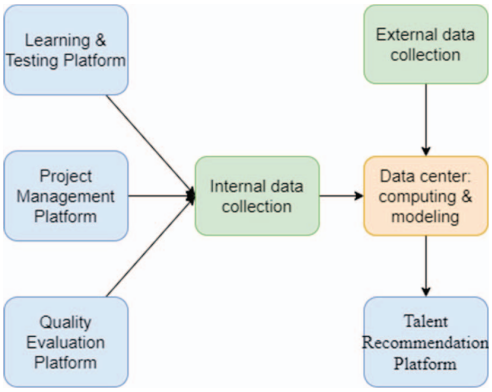

Fig 5 Platforms organization

B. System implementation

Most sub platform in SERTS adopt the design mode of front-end and back-end separation. The front-end mainly uses Vue.js's MVVM framework to achieve two-way data binding, and the back-end services mainly use Java Spring Cloud microservices framework for architecture.

To avoid cross-domain resource sharing (CORS) and cross-site request forgery (CSRF) problems arising from multi-platform communication and to enhance the scalability of the platform, Token authentication based on OAuth2.0 is used for user authentication; SERTS also uses RabbitMQ to help the internal and external data communication, and caches the necessary data (token for example) using Redis database; Spring Cloud's core component Eureka is used to facilitate the management and monitoring of multiple microservices and Zuul is used as the gateway. The Technical architecture is shown in fig 6:

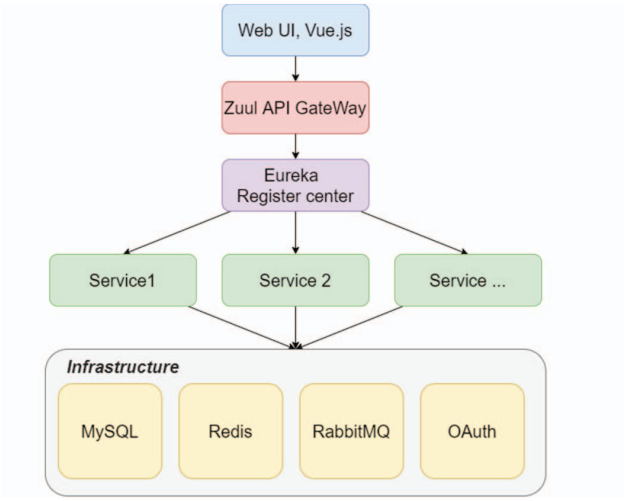

Fig 6 Technical architecture

V. Conclusion

This paper presents SERTS, a talent recommendation system in a microservice architecture, which offers a multidimensional evaluation certification for school students, and provides a good reference basis for the company's recruitment through the recommendation algorithm. We compared the differences between monolithic development, microservice development and domain-driven microservice development, and describe the architecture, design thinking and key technologies of the whole system. Next step, we will do the market validation for the demand model and try to optimize the recommendation algorithm.

Acknowledgment

The paper thanks the funding of the Foundation of AnHui Educational Committee (KJ2021A1035), the Innovative provinces special funding projects of AnHui province (2021xzx002) and the teaching and research projects of Huang Shan University (2020JXYJ25).

References

[1] Rivera-Ibarra, J. G., Rodríguez-Jacobo, J., Fernández-Zepeda, J. A., & Serrano-Vargas, M. A.(2010) Competency framework for software engineers. In: Software Engineering Education and Training Conference.

Pittsburgh. pp. 33-40.

[2] Turing, A. M. (1964) Computing machinery and intelligence. In: Minds and machines, ed. Anderson, A. R., pp. 4-30.

[3] Thones, & Johannes. (2015). Microservices. Software IEEE, 32(1), 116-116.

[4] E. Evans. (2003) Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison-Wesley.

[5] Gruber, T. R. . (1993). A translation approach to portable ontology specifications. Knowledge Acquisition, 5(2), 199-220.

[6] Horrocks, Ian, Patel-Schneider, Peter, F. , & Mike. (2004). Swrl: a semantic web rule language combining owl and ruleml. http://www.daml.org/2003/11/swrl/rules-all.html.

[7] Wong, J. A. H. A. . (1979). Algorithm as 136: a k-means clustering algorithm. Journal of the Royal Statistical Society, 28(1), 100-108.