FRI-1.414-MIP-06

IMPROVING THE STUDENT ASSIGNMENT GRADING SOFTWARE SYSTEM AT THE UNIVERSITY OF RUSE[5]

Assist. Prof. Vasil Kozov, PhD
Department of Informatics and Information Technology,
University of Ruse "Angel Kanchev"
Tel.: +359 82 888 221
E-mail: vkozov@uni-ruse.bg

Abstract: *The paper describes the business analysis process for the assignment, covering and completion of student education system assignments, as well as parts of the process itself. The necessity for improving the process and creating a robust architecture for the management of historical data, as well as creating monitoring reports and systems for better control is analysed and realised. A critical part of the created software system - meaning its' inclusion to the whole university infrastructure, and synchronising the data flow through a secure message bus is described. The iterative approach to the creation of the system is presented, as well as some of the improvements – both technological, and those related to the logical business process evolution. Most common troubles and feedback are analysed and improved upon. The results of increasing student grade coverage, as well as the impact of introducing the system into the infrastructure are presented.*

Keywords: *Education, Web, Software Architecture, Databases, Business Analytics, Service Bus (RabbitMQ).*

INTRODUCTION

For the purposes of this report, the document or object that contains the students' grades will be called a "protocol" or "student record".

The legacy system for keeping track of student records at the University of Ruse had several major problems that had to be analysed and an appropriate solution for each had to be found. The main issues will be described (Gulson, Kalervo N., and Sam S., 2019, "Emerging data infrastructures and the new topologies of education policy.").

The system had no concept of periods – semesters, years or any kind of time segregation mechanism, which necessitated a biannual loop, consisting of the following actions: close access to the protocols website, create a backup of all the data up to this point, drop and recreate the database using the new data for students, curriculum disciplines and professors, open access to the website. This by itself created massive drawbacks in terms of manual data supervision, led to direct database manipulation and infrastructure synchronization issues. This was also inconvenient for the users – system was unavailable for long periods of time, in case of mistakes in past periods it was impossible to recreate the protocols so that they could be reprinted or viewed, as the only access to "old" data was through the backups of the previous databases. Another major issue was that the system for grading was not included into the university software infrastructure – there was no way in practice for a student to view their grades, as their profiles were not connected to the grading system. It was the same for administrative personnel and professors – they did not have access to the historical performance data of their students. Management also lacked the necessary tools to view and query information about the overall state of the education processes in the university, which led to manual tasks and reports that were next to impossible to automate given the way the data was stored. Lack of infrastructure connections also meant that student curriculum changes were not included to the grading system in the appropriate time-frame, as they could not be updated real-time and every time they had to be manually edited in.

In order to improve the business process, as well as the overall university software infrastructure, several general directions for the functional requirements were created and discussed

[5] Докладът е представен на конференция на Русенския университет на 28 октомври 2022 г. в секция Математика, информатика и физика с оригинално заглавие на български език: ПОДОБРЯВАНЕ НА СОФТУЕРНАТА СИСТЕМА ЗА ОЦЕНЯВАНЕ НА СТУДЕНТСКИТЕ РАБОТИ В РУСЕНСКИЯ УНИВЕРСИТЕТ.

with the relevant parties (Haigh, M., 2010, "Software quality, non-functional software requirements and IT-business alignment.").

EXPOSITION

Lack of control tools and monitoring reports led to the concentration of managing power into people that should not have the access or responsibility to manipulate such data. A solution for the division of responsibilities was discussed and implemented into the model, aiming at the correct distribution of obligations and access to data. It took the form of active directory groups that have certain levels of access and manipulation of data and user (professor) access. After the recognition of the aforementioned issues and difficulties in the business process and the software, the method for dealing with the problem was described and implemented. It was decided that in order to decentralize the responsibilities and significantly reduce errors, the administration of each unit had to be handled by the units themselves. As each faculty in the university has multiple units for handling different sciences and their respective curriculum subjects, the people responsible for the study process are the most knowledgeable about their current curriculum state. As such, these people are the most qualified to accurately signal for issues with their respective curriculums, and have the knowledge on which professor is responsible for the subjects they teach. A special role was introduced for them, called unit administrator, meaning the power to control which professor can access which curriculum discipline was given to the units, thus improving the distributed characteristic of the process and diminishing its centralization. All roles can be observed on Figure 1.

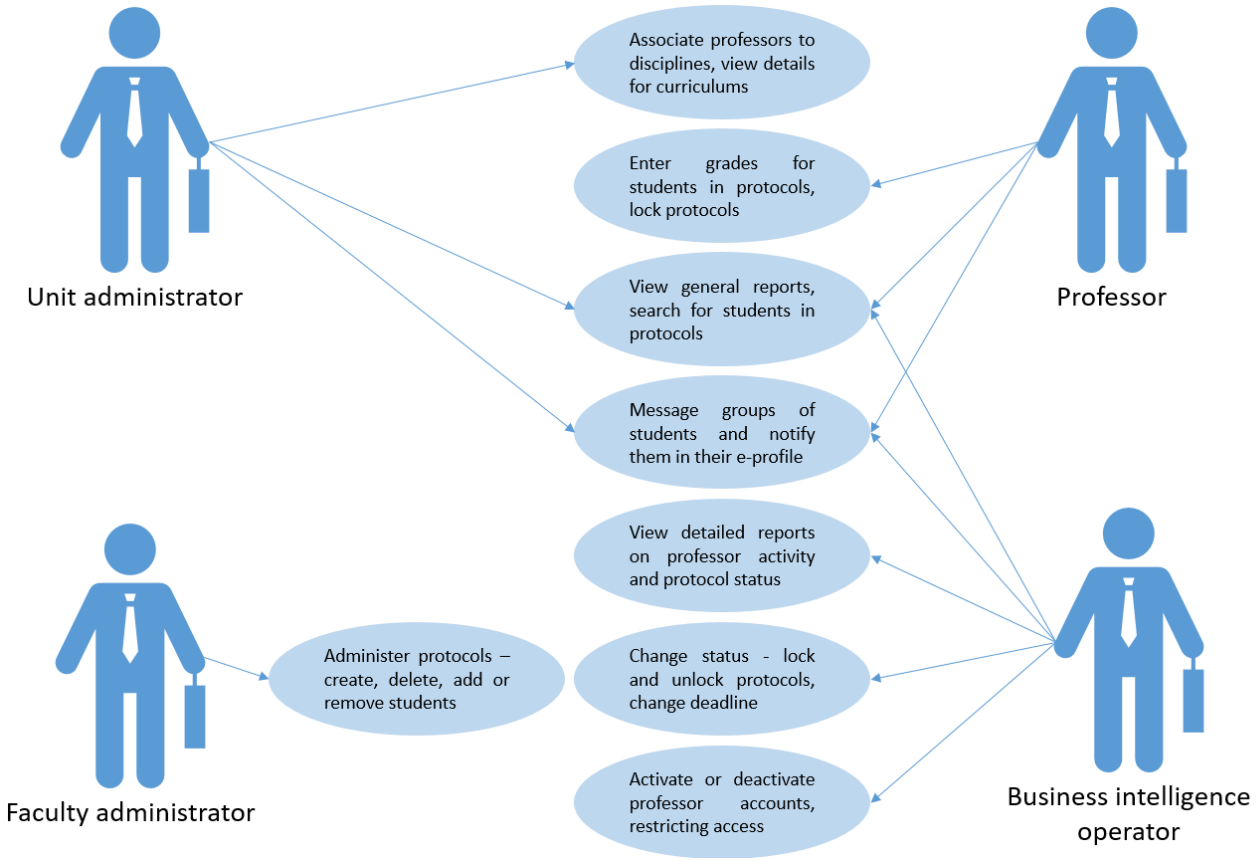

Figure 1. Use case for some of the main functionalities of the software system and the user roles that have access to them

The other roles – such as the faculty administration personnel, as well as managing personnel and business intelligence groups – were defined in detail, so that every role has an accurate description. Their responsibilities were generalized and the business process diagrams for its use cases were formed.

Having analyzed the use cases and clarified the business process, the next important step was the choice of technologies, that would be used to develop and integrate the system.

In order to successfully include the system to the already existing infrastructure, the technological stack must be compatible (Garcia, J., Mehdi M., Lu X., et al., 2021, "Constructing a shared infrastructure for software architecture analysis and maintenance."). As most of the integrated software has been developed in the Microsoft .NET stack, and the skill requirements in working with it were present, it was decided to follow the established pattern and choose a new iteration of a Microsoft technology to fit in with the rest of the systems. The business cases require a solution in the web, as it would be impractical to make a mobile or desktop application, and a cloud solution is unfeasible due to the already existing servers we can use. Following these simple conclusions, the choice was easy - .NET framework. At the time of the creation of the software system, the current version of .NET framework was 4.5.2, and it was upgraded at a later date iteratively.

As for the user interface (UI), the usual suspect – bootstrap 3, was chosen. In one of the following agile iterations during the visual overhaul of the system it was substituted with its' newest available bootstrap 5 library. There were significant changes between the library versions that provided more options and streamlined the interface creation, thus allowing improvement in UI versioning and future changes (Figure 2).

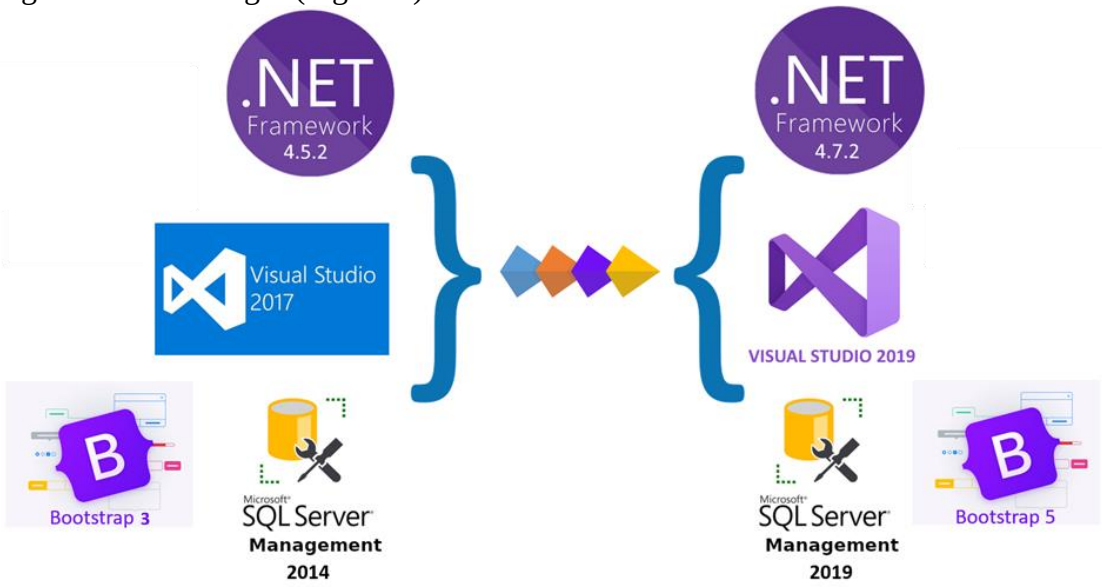

Figure 2. Change of some of the technologies used during the development and implementation of the project

Active Directory Federation Services were used for login authentication, groups, corresponding with the actors in the use case diagram were created and filled for the authorization to different functionalities of the system. As the university-wide authentication system for the entire infrastructure, the technology is robust and integrates well with the Microsoft stack.

Integration into the software infrastructure requires communication with services and applications that are handling student data, curriculums and obligations. In order to solve this issue, a service bus approach was targeted, as a similar approach had already been used between existing systems, using a locally deployed legacy Microsoft Service Bus. Expert opinion was sought from software architect specialists working in the private sector about the relevance of such an approach. Their positive confirmation was accompanied with a recommendation to use a locally deployed RabbitMQ as a service bus. After reading the documentation and reliable sources (Ionescu, V., 2015, "The analysis of the performance of RabbitMQ and ActiveMQ."; Rostanski, M., Krzysztof G., and Aleksander Seman, 2014, "Evaluation of highly available and fault-tolerant middleware clustered architectures using RabbitMQ."), a working solution was tested, the bus was deployed on a local Windows server, and test communication was established. A simplified model of the way the service bus works can be seen on Figure 3.

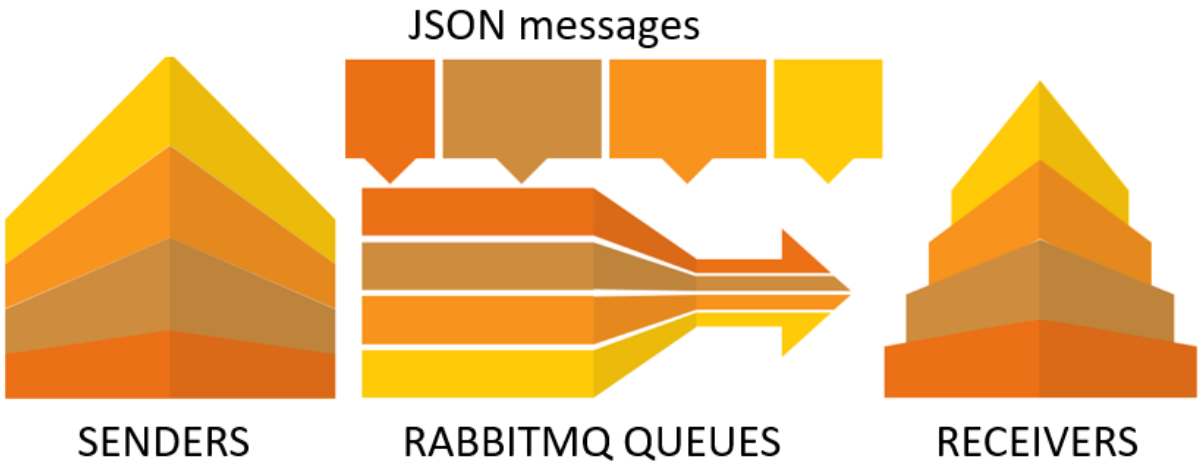

Figure 3. Simplified model of a working message bus

The system receives and sends JSON messages to and from the other infrastructure software services using specialized queues for each domain object of concern.

The next step after the analysis was made and the technologies were chosen, was the implementation itself. The beginning was marked by the design and modeling of the database that would host the necessary data for the correct functioning of the system. In order to develop an initial version, the agile software approach was used (Al-Saqqa, S., Samer S., and Hiba A., 2020, "Agile Software Development: Methodologies and Trends."; Hoda, R., Norsaremah S., 2018, "The rise and evolution of agile software development."). It was slightly modified over the time of the exploitation of the system, as changes to the business process necessitated corrections. The final version of the model of the database at the moment of the publishing of this article can be seen on Figure 4.

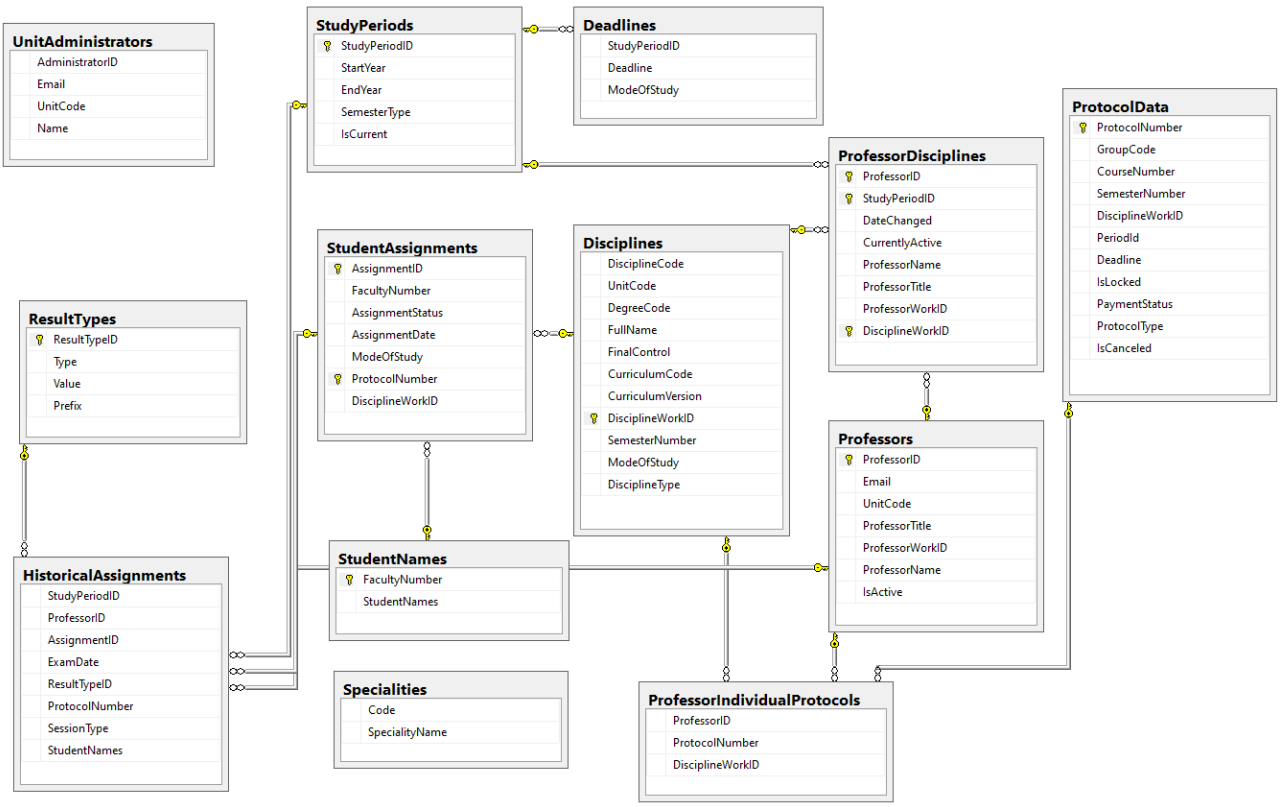

Figure 4. Model of the relational database of the system

Having defined the domain objects which help describe and facilitate the business processes, the MVC architecture was used to create a web application backed by API and services, deployed on

a windows severs' IIS. Through continuous testing and development iterations (Figure 5), the old software system was phased out, and the new one took its place, connecting the process of assigning grades to student assignments available throughout the entire university infrastructure. The communication through the new message bus proved far easier than what was achieved previously. Swapping currently deprecated technologies with new but tested ones proved a challenging, but interesting and enriching process.

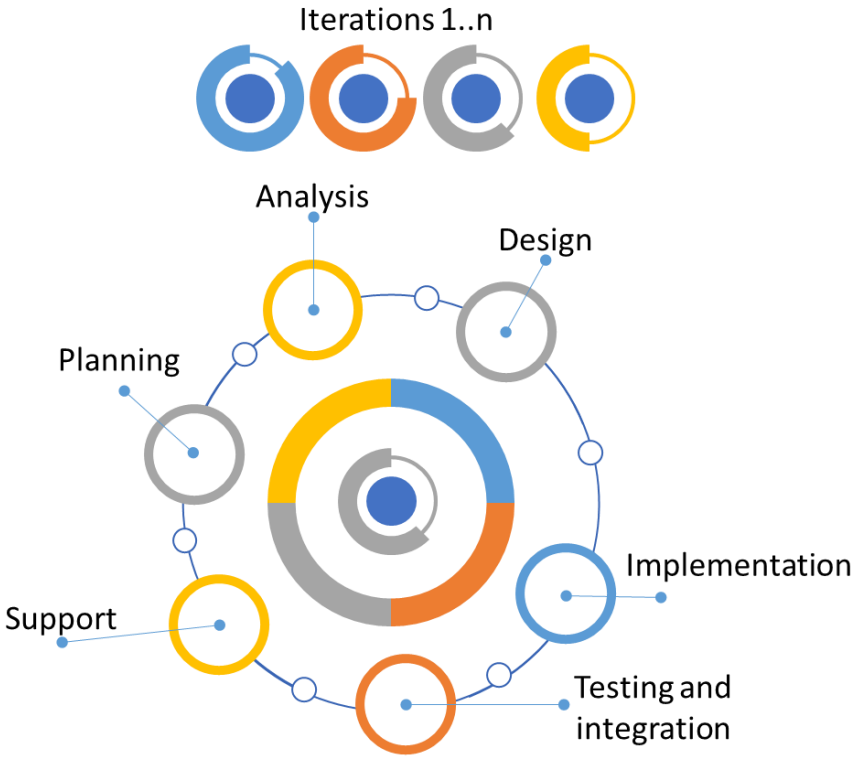

Figure 5. Continuous testing and development iterations

In order to support the agile development process, Team foundation server was used for hosting the code, the tasks and the versioning of the project. Having tasks associated to parts of the code versions was critical in finding bugs throughout the iterations. For tasks that did not require coding and were more akin to quick notes, Trello (by Atlassian) was used for team communication. It proved useful having a quick way to access brainstorming boards with inspiring little tasks, as well as having the creative ideas separated from the heavy logical use cases and user stories in the TFS.

CONCLUSIONS

The introduction of the software system in practice has led to increased clarity of the business processes around completing and modifying student assignments for all participating parties, a balanced division of responsibilities, less manual work, a better reporting system as a tool for better management and control, the ability for students to view their accomplishments in real time, as well as increased sense of professor responsibility.

The software system has been continuously improved upon using the iterative development process successfully (Tam, C., Eduardo J., Tiago O., 2020, "The factors influencing the success of on-going agile software development projects."), different versions have rolled out for the users throughout its' exploitation period, and there are features that are planned for the future that will further lessen the cognitive load on the unit administrators. Feedback is taken into account when designing new and editing already existing functionalities, which leads to a positive dialogue between the developers and the users. Future development includes the integration of individual protocols, and even more detailed reports, helpful for data analysis on the management level.

REFERENCES

Al-Saqqa, Samar, Samer Sawalha, and Hiba AbdelNabi. "Agile Software Development: Methodologies and Trends." International Journal of Interactive Mobile Technologies 14, no. 11 (2020).

Garcia, Joshua, Mehdi Mirakhorli, Lu Xiao, Yutong Zhao, Ibrahim Mujhid, Khoi Pham, Ahmet Okutan et al. "Constructing a shared infrastructure for software architecture analysis and maintenance." In 2021 IEEE 18th International Conference on Software Architecture (ICSA), pp. 150-161. IEEE, 2021.

Gulson, Kalervo N., and Sam Sellar. "Emerging data infrastructures and the new topologies of education policy." Environment and Planning D: Society and Space 37, no. 2 (2019): 350-366.

Haigh, Maria. "Software quality, non-functional software requirements and IT-business alignment." Software Quality Journal 18, no. 3 (2010): 361-385.

Hoda, Rashina, Norsaremah Salleh, and John Grundy. "The rise and evolution of agile software development." IEEE software 35, no. 5 (2018): 58-63.

Ionescu, Valeriu Manuel. "The analysis of the performance of RabbitMQ and ActiveMQ." In 2015 14th RoEduNet International Conference-Networking in Education and Research (RoEduNet NER), pp. 132-137. IEEE, 2015.

Rostanski, Maciej, Krzysztof Grochla, and Aleksander Seman. "Evaluation of highly available and fault-tolerant middleware clustered architectures using RabbitMQ." In 2014 federated conference on computer science and information systems, pp. 879-884. IEEE, 2014.

Tam, Carlos, Eduardo Jóia da Costa Moura, Tiago Oliveira, and João Varajão. "The factors influencing the success of on-going agile software development projects." International Journal of Project Management 38, no. 3 (2020): 165-176.