Enhancing Student Mobility in Higher Education: A Case Study on Application Architecture and Digital Services Supporting Processes and Transactions

Vjeran Strahonja, Katarina Pažur Aničić, Izabela Oletić Tušek
University of Zagreb, Faculty of Organization and Informatics, Varaždin, Croatia
vjeran.strahonja@foi.unizg.hr, kpazur@foi.unizg.hr, izabela.oletic@foi.unizg.hr

DOI
https://doi.org/10.18690/um.fov.2.2025.65

ISBN
978-961-286-963-2

Enabling student mobility within higher education involves complex, interconnected processes that are supported by various digital tools and applications. This study focuses on these processes at the University of Zagreb, Faculty of Organization and Informatics, conducting a detailed case study to analyse the existing digital services that facilitate student mobility. It aims to define and discuss the application architecture of an integrated system and explore how these services can be further developed by integrating legacy components, open services, and open-source elements. The proposed application architecture should support basic transactions, workflows and the need for data storage and analysis, thereby enhancing system interoperability and simplifying processes for all stakeholders involved—including students, international relations officers, and academic advisors. By proposing further digitalisation, this paper contributes to simplifying and improving of the mobility process, making it more accessible and efficient for the international education community.

Keywords:
student mobility,
digital services,
platform and application,
system and architecture,
case study

University of Maribor Press

1 Introduction

Digitalisation has influenced organizations across various sizes and sectors, including higher education (HE), which is no exception. Digitalisation (known as well as digital transformation) involves the development and implementation of digital tools and services aimed at improving processes, focusing both on leveraging digital technologies and enhancing user experience (Henriette et al., 2016). While digitalisation offers numerous benefits to organisations, introducing diverse information systems (IS) also presents challenges. Over time, these systems often require redesigns, functional improvements, updates to their architecture or technology, and integration with other IS, ensuring interoperability at different levels (European Commission, 2017).

Fernández et al. (2023) highlight that digital transformation initiatives in education primarily focus on delivering quality and competitive education (24% of 184 initiatives across 39 universities). However, 16% of efforts address optimising information security and maintaining business continuity, including secure information sharing and centralising services and data sources. Hassani and Mansouri (2017) point out that digitalisation can create challenges, such as data desynchronisation across university systems, forcing users to re-enter the same information in multiple platforms—an issue that increases the risk of errors. For higher education institutions (HEIs), establishing reliable processes while safeguarding data is essential. These challenges are particularly critical in the higher education ecosystem, where effective data exchange is essential. Student mobility is one of the key aspects where such issues are highly significant, as the credit mobility process involves numerous interconnected processes and stakeholders (Tomičić Furjan et al., 2022), requiring seamless coordination among entities such as the home institution, the home university (if applicable), and the host institution.

This study focuses on student mobility processes at the University of Zagreb, Faculty of Organization and Informatics (UNIZG FOI), conducting a case study to analyse the existing digital services that facilitate student mobility. It aims to define and discuss the application architecture of an integrated student mobility support system (SMSS) and explore how these services can be further developed through the integration of legacy components, open services, and open-source elements.

2 Student mobility processes at the University of Zagreb, Faculty of Organization and Informatics: a digital transition

The student mobility process often requires navigating various digital systems and platforms. At the University of Zagreb, the application procedure has been digitalized through the "Move On" platform. This system allows students to submit a single application form, which includes all required documents uploaded as attachments. However, the application form itself must be printed, signed, and subsequently submitted in its final form (to the international office of the student's faculty). Once (digitally) submitted, the application cannot be modified. Failure to follow all procedural steps—such as submitting multiple applications, excluding required documents or late submission—results in automatic rejection. Following the announcement of results, the International Relations Office at the University of Zagreb faculties nominates students for a semester at the host institution, often through digital platforms. While email was the primary tool for student nominations a few years ago, its usage is gradually being phased out in favour of more automated systems, reducing personal interaction between international offices.

Upon nomination, students typically receive detailed email instructions regarding the application process at the host institution. This communication often includes a fact sheet or information guide outlining essential details, particularly for the Erasmus Without Paper (EWP) Dashboard and the Online Learning Agreement (OLA). Students may be required to upload a signed PDF version of the OLA from their home institution. Alternatively, some host institutions mandate using their internal platforms for course selection, supplementing the OLA. Accommodation applications are frequently integrated into these systems, though in some cases, they are handled separately.

During their mobility period, students utilise the EWP Dashboard to make any necessary changes to their learning agreements. Upon concluding their mobility, students often engage with additional digital platforms to access grade transcripts or transcripts of records. Increasingly, these documents are distributed through online systems rather than traditional email communication. Upon return, students from the University of Zagreb's Faculty of Organization and Informatics utilize the FOI Forms platform to submit grade transcripts and other documentation for course recognition. The final set of required documents is then emailed to the University's central International Relations Office.

To illustrate the overlap in the data that students must re-enter across different systems during the mobility application process, the authors analysed the application form within the MoveON platform (Sveučilište u Zagrebu, 2024) and the Digital Erasmus+ Learning Agreements within the EWP platform (European Commission, 2023). Students must repeatedly enter the same information into both systems, including:

- **Student identity**: First name, Last name, Birth date, Gender, Nationality, Field of education (ISCED) / ISCED code, Level of education
- **Receiving institution**: Name (Institution), Faculty/Department, Country
- **Language**: Language of instruction/Teaching language, Language competence
- **Estimated duration**: From – month, From – year, To – month, To – year

Additionally, within EWP, students must provide details about administrative and responsible persons at the host institution. They are also required to input comprehensive data related to the learning agreement, including Courses at the Host Institution (Component code, Component title, Term, Number of ECTS credits (or equivalent) to be awarded upon successful completion, and Web link to the course catalogue describing the learning outcomes) and Courses at the Sending Institution (Component title, Term, Number of ECTS credits, Automatic recognition, and Provisions for incomplete educational components).

This redundancy and manual entering of data available in different systems not only complicates and prolongs the application process but also increases the potential for errors, underscoring the need for better integration and synchronisation of these systems.

2.1 Problem domain

The development of a student mobility support system (SMSS) requires a well-designed architecture that covers core functionalities and ensures openness and interoperability with other systems, such as student records, university enterprise resource planning (ERP), e-learning platforms, etc. A proposal for core functionalities, following the main processes of student mobility as performed in practice, is below:

a) Stakeholder register and contact management
- Records of partner institutions and contact persons
- Connecting stakeholders to mobility programs
- Management of bilateral agreements and their conditions

b) Program and call management
- Registry of programs (Erasmus mobility for studies, short intensive programmes, bilateral cooperation, internships, etc.)
- Connecting programs with stakeholders (institutions, agencies, companies)
- Generating and managing calls (deadlines, conditions, documentation)
- Monitoring student applications throughout the entire workflow (application, selection, evaluation)

c) Student mobility management
- Monitoring applications and mobility status for each student
- Automatic allocation according to transparent criteria (equality, non-discrimination)
- Generation of mobility contracts with customisable items
- Monitoring of cash flows and calculation according to grant schemes

d) Recognition of achievements
- Registry of equivalences between incoming and outgoing institutions
- Automation of the process of recognition of achieved achievements (ECTS credits, grades)
- Interoperability with existing academic systems

e) Information portal for students
- Chatbot for support and FAQ (e.g. for information on competitions, deadlines, conditions)
- Thematic forums and groups for the exchange of experiences among students
- Notifications and personalised information system

f) Financial monitoring and reporting
- Records of mobility-related transactions (scholarships, grants)

- Automation of calculations according to grant schemes
- Preparation of financial reports

g) Interoperability with the study support system

- Integration with existing systems for recording students, courses, and grades (e.g. ISVU or similar systems)
- Data exchange through standardised protocols (e.g. EWP - Erasmus Without Paper).

3 Methodological framework for building a system based on the integration of legacy and new components

The case discussed in this paper (Student Mobility Support System – SMSS) is typical for all business problem domains and all organisations that have been operating for a certain time and have some kind of legacy information systems. A significant redesign or improvement of the functionality, architecture or technology of an information system is a complex undertaking that requires an appropriate methodological framework (Sommerville, (2020; Wieringa, 2014), i.e. a structured description of the approaches, principles, processes, methods, techniques and tools used to solve a particular class of problems in some scientific or professional field, together with instructions for their application in various problem situations. An important part of the methodological framework is the development cycle and process patterns, which describe which phases, processes, and activities comprise the development cycle, how they are interrelated and which are conditions for their initiation or completion. The life cycle is a broader concept because, in addition to the phases of the development cycle, it includes the planning phases that precede the development, and the implementation, use, maintenance and continuous improvement of the system.

Different scenarios for building such a system are possible:

1. Finding a new ready-made parameterised system that entirely or to a considerable extent solves existing problems and satisfies needs;
2. Building a new custom system based on your own requirement specifications;

3. Improving or replacing existing "legacy" application components, and integrating with ready-made or customised components that cover missing functionalities.

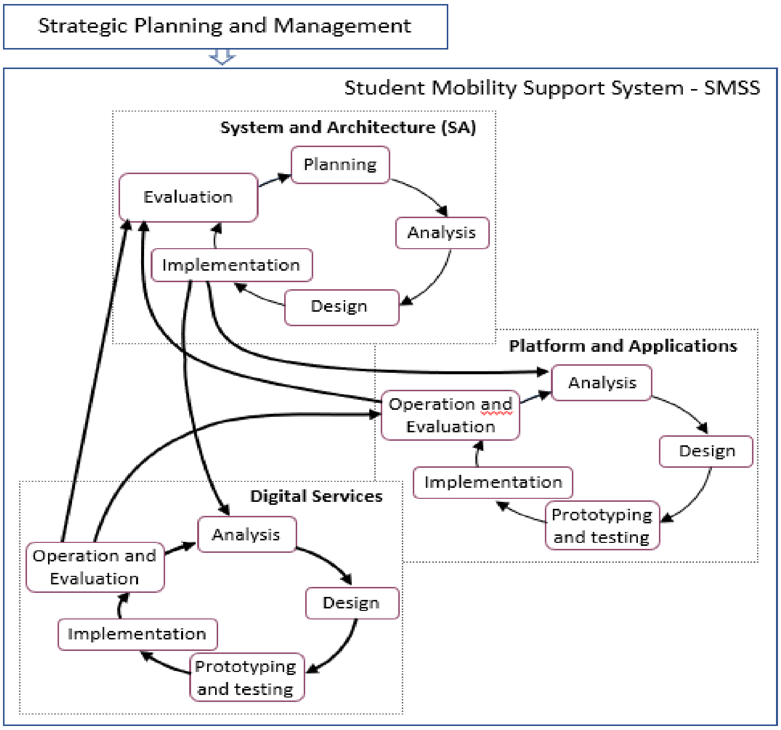

Figure 1: Life cycle applicable to the Student Mobility Support System
Source: Own

The first and second scenarios are relatively rarely encountered in practice today. The third scenario is likely. For development based on the improvement of existing components and integration with new components, a development cycle, as shown in the Figure 1, is proposed. The initiation of the new development cycle is in the area of Strategic Planning and Management, where the influence of computer applications, digital services and other technologies on the achievement of the organization's strategy and business model is analysed, feasibility is studied, and strategic projects related to IS are planned and launched.

The Figure 1. shows that the proposed life cycle is not monolithic, but incorporates three interconnected life cycles: System and Architecture, Platform and Applications and Digital Services. Each of them proceeds through several phases.

The System and Architecture (SA) life cycle includes the phases in which the SMSS as a whole is conceived and planned and provides the methodological and architectural framework for the platform, application and services. Its Planning phase includes defining the project's purpose, objectives and scope, success indicators, project structure and time frame, resources, etc.

The Analysis phase of the SA life cycle has two groups of activities (Gunawardhana, 2019; Catanio, 2006):

1. Requirements Analysis
   - Contextual requirements analysis – identifying and analysing requirements arising from strategies, policies, business models and other higher-level documents, legal and regulatory requirements, etc.
   - Business process and functionality identification – describing and analysing "as-is" business processes, data sets, services and other functionalities covered by the legacy system and identifying missing or inadequate (pains and gains analysis, etc.).
   - Technical analysis – analysing the existing architecture, infrastructure, platforms, applications and other system components and defining possible improvements in the existing context
   - Defining functional and non-functional requirements – defining the properties and requirements the "to-be" system must meet (functionality, data, performance, security, scalability, user interface, etc.)
2. Evaluation and Selection
   - Research of available components – identifying and analysing existing legacy components and available new components that meet functional requirements (open-source, commercial off-the-shelf – COTS), where platforms such as GitHub, SourceForge and Docker Hub can be useful.

- Evaluation of potential solutions – mapping to functional and non-functional requirements and comparing components based on functionality, adaptability, licensing, security, community support, etc.
- Gap analysis – identification of functional and non-functional requirements that are not covered by any component and defining how to satisfy them, including development based on the requirements.
- Technical evaluation – technical verification, considering the results of the technical analysis and compatibility with the legacy system.
- Proposal of technical options – make or buy analysis, selection and argumentation of the architecture and components that best meet the requirements.

The analysis involves intensive iteration with relevant stakeholders to ensure that the requirements specification includes various aspects and comprehensive, relevant information.

Design at the SA level includes:

- Architecture and integration design - listing legacy and new system components, defining the integration of different applications and other components to communicate and exchange data using APIs, middleware, microservices, message brokers, adapters, etc.
- Data design - logical design of a standard data model
- Adaptation planning - defining the adaptation of each legacy and new component.
- Orchestration and integration planning - defining the management of the order and manner in which different components interact to achieve more complex processes or workflows.

The Implementation of what is planned, analysed and designed in the SA life cycle is actually realised in the life cycles of Platforms and Applications and Digital Services.

The Platform and Applications lifecycle is common for application systems and includes the phases of Analysis, Design, Prototyping and Testing, and Implementation, which are the usual phases of the software development cycle. Operation and Evaluation of Platform and Applications includes their maintenance and upgrades, as is common for applications and information systems.

The Digital Services lifecycle may include the phases shown in the Figure 1, or any other arrangement of phases according to some service development methodology.

Although it works on three life cycles with distinct phases, development in each of the life cycles can be performed incrementally and iteratively, according to agile principles. If evaluation during operational use of a platform, application or digital service indicates that significant changes are needed, a new development cycle for that resource is initiated either at the System and Architecture level.

4 Selected methods of analysis, evaluation and selection of application components

The chosen approach to building SMSS is improving or replacing existing legacy application components and integrating them with ready-made or customised components that cover missing functionalities. The relevant literature describes different methodological approaches, which are discussed below.

Reference application models at an abstract level represent the structure, functionality, and behaviour that can be considered a standard or a good reference practice in a particular application area. They serve as a guide or foundation for understanding and structuring complex systems, defining requirements, comparing and evaluating applications, aligning with industry standards, etc. (Scheer et al. 2002). It is important that there are methods for similarity measurements of application component models and reference modls expressed by using UML diagrams (Triandini et al. 2022; Adamu et. al. 2019). Metamodels are "models of models", i.e. abstract representations of concepts, structures and rules for building specific models. They define the language and rules for creating specific models. In software architecture, they facilitate the understanding and specification of model concepts and their interrelationships. The already mentioned Triandini et al. (2022) research is also useful in determining the similarity of metamodels to each other and metamodels and models. Ontologies are formal specifications of knowledge domains, which define, structure and classify concepts, their properties and relationships between them. They enable a consistent and precise interpretation of terms in an application domain and an automated search and selection of software solutions based on a semantic understanding of the problem, as well as analysis of enterprise architecture (EA) models, particularly addressing the challenges of syntax, structure, and semantic heterogeneities (Bakhshandeh et al. 2016).

Table 1. summarizes the advantages and disadvantages of using reference models, metamodels, and ontologies for evaluating application components in the context of modernizing a legacy system and integrating new components.

Table 1: Comparison of reference models, metamodels, and ontologies for evaluating application components

Approach	Advantages	Disadvantages
Reference Models	- provide insight into best practices and standards in an application area - facilitate design, compliance verification, evaluation and component selection by aligning with proven approaches - help identify gaps in functionality or compliance.	- they are generic in nature - are not flexible and require efforts in tailoring, supplementing and adapting to domain-specific or highly customized requirements - reflect existing and may lag behind technological advances.
Metamodel s	- provide a high-level abstraction for understanding the structure and relationships of application components - support systematic comparison and evaluation of components - can support architectural consistency and model-driven development (MDD)	- require time and expert knowledge to develop, interpret and apply effectively, - they are focused on structure rather than functionality and behavior, - it can be difficult to map to concrete implementations in legacy systems.
Ontologies	- provide a formalism for the definition, representation, classification and understanding of concepts, relations and rules in the domain, - enable semantic reasoning to identify overlaps, dependencies, and inconsistencies among components, - support semantic interoperability and integration of heterogeneous components.	- development of domain-specific ontologies is time-consuming for complex applications, - require significant expertise in both domain and ontology design, - there is a gap in the implementation of ontologies - difficulties in integrating with legacy systems not designed for semantic structures.

5 Conclusion

This paper presents the authors' initial steps in designing a digital Student Mobility Support System (SMSS), based on the case study from the UNIZG FOI. The presented Methodological framework set a basis for further endeavours in creating a sustainable SMSS aimed at supporting and simplifying student mobility process.

Acknowledgment

This work was supported by the project "SuMoS - Strengthening the ecosystem for sustainable student mobility", financed from the Erasmus+ Programme of the European Union, within KA220-HED – Cooperation partnerships in higher education (2024-1-HR01-KA220-HED-000254853). The sole responsibility for the content of this article lies with the authors. It does not necessarily reflect the opinion of the European Union.

References

Adamu, A., Zainon, W. M. N. W., & Abdulrahman, S. M. (2019). Empirical investigation of UML models matching through different weight calibration. *In Proceedings of the 2019 8th International Conference on Software and Computer Applications* (pp. 1672234323323-172).

Bakhshandeh, M., Pesquita, C., & Borbinha, J. (2016). An ontological matching approach for enterprise architecture model analysis. *In Business Information Systems: 19th International Conference, BIS 2016,* Leipzig, Germany, July, 6-8, 2016, Proceedings 19 (pp. 315-326). Springer International Publishing.

Catanio, J. T. (2006). Requirements analysis: A review. *Advances in Systems, Computing Sciences and Software Engineering: Proceedings of SCSS05*, 411-418.

European Commission. (2023) Erasmus+: Comprehensive User Guide for Digital Erasmus+ Learning Agreements. https://erasmus-plus.ec.europa.eu/sites/default/files/2023-10/User-Guide-Erasmus-Digital-LA_en.pdf

European Commission: Directorate-General for Digital Services. (2017). New European interoperability framework : promoting seamless services and data flows for European public administrations. Publications Office. https://data.europa.eu/doi/10.2799/78681.

Fernández, A., Gómez, B., Binjaku, K. et al. (2023.) Digital transformation initiatives in higher education institutions: A multivocal literature review. *Educ Inf Technol* 28, 12351–12382. https://doi.org/10.1007/s10639-022-11544-0

Gunawardhana, L. P. D. (2019). Process of requirement analysis link to software development. *Journal of Software Engineering and Applications*, 12(10), 406.

Hassani A.L., Mansouri, K. (2017) Towards the development of middleware architecture, interoperability of university's information systems integrating online learning platforms, *INTED2017 Proceedings,* pp. 4528-4537.

Henriette, E., Feki, M., and Boughzala, I., Digital Transformation Challenges (2016). In *Mediterranean Conference on Information Systems MCIS 2016* Proceedings. 33.

Scheer, A. W., & Nüttgens, M. (2002). ARIS architecture and reference models for business process management. *In Business Process Management: Models, Techniques, and Empirical Studies* (pp. 376-389). Berlin, Heidelberg: Springer Berlin Heidelberg.

Sommerville, I. (2020). *Engineering software products* (Vol. 355). London: Pearson.

Sveučilište u Zagrebu (2024) UPUTE za ispunjavanje ON-LINE PRIJAVE za NATJEČAJ za mobilnost studenata u svrhu studijskog boravka u okviru Erasmus+ programa ključne aktivnosti 1 u akademskoj godini 2024./2025. https://www.unizg.hr/fileadmin/rektorat/Suradnja/Medunarodna_razmjena/Studenata/Erasmus_SMS/2024_25/UPUTE_za_ispunjavanje_ONLINE_prijave_2024_25.pdf

Tomičić Furjan, M., Kurešević, M., Oletić Tušek, I. (2022) International mobility – case of student exchange management, *EDULEARN22 Proceedings*, pp. 3803-3809.

Triandini, E., Fauzan, R., Siahaan, D. O., Rochimah, S., Suardika, I. G., & Karolita, D. (2022). Software similarity measurements using UML diagrams: A systematic literature review. *Register: Jurnal Ilmiah Teknologi Sistem Informasi*, 8(1), 10-23.

Wieringa, R. J. (2014). *Design science methodology for information systems and software engineering.* Springer

About the authors

Vjeran Strahonja, Ph.D. is a Full Professor at the UNIZG FOI. The field of his scientific and teaching work is system and software engineering, IT service management and service engineering.

Katarina Pažur Aničić, Ph.D. is an Associate Professor at the UNIZG FOI. The field of her scientific and teaching work is IT service management and project cycle management. Currently, she is the Faculty ECTS coordinator and academic advisor for exchange students.

Izabela Oletić Tušek has been the Head of the International Relations Office at the UNIZG FOI since 2010. She has extensive experience in internationalization processes and cooperation.