UDC 004.42
DOI https://doi.org/10.32689/maup.it.2025.3.22

Yurii STATYVKA
Candidate of Technical Sciences, Associate Professor,
Associate Professor at the Department of Software Engineering in Energy,
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute",
yu.statyvka@gmail.com
***ORCID:** 0000-0002-7734-953X*

Zhang MINGJUN
Postgraduate Student at the Department of Software Engineering in Energy,
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute",
dora_ZMJ@163.com
***ORCID:** 0000-0002-7189-2881*

SOFTWARE DESIGN TECHNOLOGY FOR AUTOMATING THE PROCESS OF EVALUATING THE LEVEL OF INTERNATIONALIZATION OF SCIENTIFIC INSTITUTION'S ACTIVITIES

Abstract. *The article is devoted to the development of software design technology for automating the process of evaluating the level of internationalization of a scientific institution's activities by developing the IRI-Frame architectural template, which differs from universal templates in that it is focused on the specifics of internationalization assessment.*

The aim of this paper. *Development of software design technology for automating the evaluation of the level of internationalization of scientific institutions' activities based on the IRI-Frame architectural template and system models that support the collection, processing and publication of data for expert decision-making.*

Methodology. *An analysis of the subject area and system requirements was conducted, and an IRI-Frame architectural template was created, focused on the specifics of internationalization evaluation. The static model describes the main artifacts (project, methodology, specification) and their interaction through the Designer, Specifier, Choicer, Estimator, Publisher classes, which inherit the IRIS functionality. The dynamic model reflects scenarios – from the creation and study of methodologies to the selection of the best one and the publication of results. The technology includes three stages: environment definition, requirements formulation, and design of universal and specialized components.*

Scientific novelty. *An IRI-Frame architectural template is proposed, which isolates universal components (support for system creation and evaluation) from special ones (methods of computation, data access, results publication). The combination with GoF patterns provides configuration changes without rewriting the code. The developed static and dynamic models detail the processes of choosing methods of normalization, aggregation, weighting, and statistical analysis.*

Conclusions. *The use of IRI-Frame simplifies the development of internationalization evaluation systems, ensures scalability and integration with various data sources. The technology is suitable for creating separate systems and integrated components. The models reflect the logic of work from project initiation to results publication, forming the basis for further expansion of evaluation methods and optimization of algorithms.*

Key words: *internationalization of scientific institutions, evaluation of the level of internationalization, architecture of the software system, software engineering.*

Юрій СТАТИВКА, Чжан МІНЦЗЮНЬ. ТЕХНОЛОГІЯ ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ ПРОЦЕСУ ОЦІНКИ РІВНЯ ІНТЕРНАЦІОНАЛІЗАЦІЇ ДІЯЛЬНОСТІ НАУКОВОЇ УСТАНОВИ

Анотація. *Стаття присвячена розробленню технології проєктування програмного забезпечення для автоматизації процесу оцінки рівня інтернаціоналізації діяльності наукової установи за допомогою розробки архітектурного шаблону IRI-Frame, який відрізняється від універсальних шаблонів тим, що орієнтований на специфіку оцінювання інтернаціоналізації.*

Мета роботи. *Розроблення технології проєктування програмного забезпечення для автоматизації оцінювання рівня інтернаціоналізації діяльності наукових установ на основі архітектурного шаблону IRI-Frame та моделей системи, що підтримують збір, обробку й публікацію даних для прийняття експертних рішень.*

Методологія. *Проведено аналіз предметної області та вимог до системи, створено архітектурний шаблон IRI-Frame, орієнтований на специфіку оцінювання інтернаціоналізації. Статична модель описує основні артефакти (проєкт, методологія, специфікація) та їх взаємодію через класи Designer, Specifier, Choicer, Estimator, Publisher, що успадковують функціонал IRIS. Динамічна модель відображає сценарії – від створення та дослідження методологій до вибору найкращої та публікації результатів. Технологія охоплює три етапи: визначення середовища, формування вимог і проєктування універсальних та спеціалізованих компонентів.*

Наукова новизна. *Запропоновано архітектурний шаблон IRI-Frame, що ізолює універсальні компоненти (підтримка створення та оцінювання систем) від спеціальних (методи обчислень, доступу до даних, публікації* *результатів). Поєднання з патернами GoF забезпечує зміну конфігурацій без переписування коду. Розроблені статична та динамічна моделі деталізують процеси вибору методів нормалізації, агрегування, зважування та статистичного аналізу.*

Висновки. *Використання IRI-Frame спрощує розробку систем оцінювання рівня інтернаціоналізації, забезпечує масштабованість і інтеграцію з різними джерелами даних. Технологія придатна для створення окремих систем та інтегрованих компонентів. Моделі відображають логіку роботи від ініціювання проєкту до публікації результатів, формуючи основу для подальшого розширення методів оцінювання й оптимізації алгоритмів.*

Ключові слова: *інтернаціоналізація діяльності наукових інституцій, оцінювання рівня інтернаціоналізації, архітектура програмної системи, інженерія програмного забезпечення.*

Introduction. The concept of internationalization of scientific activity emerged in the last decades of the last century as a result of the process of globalization of the world economy. The concepts of internationalization and globalization are not identical or even synonymous, although they are sometimes used interchangeably. A more balanced approach assumes that internationalization implies the presence of nations and nation-states and their movement towards interaction with other nations, cultures, etc. in the context of scientific activity, while globalization is simply "the flow of technologies, economy, knowledge, people, ideas across borders" [5].

Measuring the level of internationalization of scientific activity is a difficult problem due to the complexity and diversity of the phenomenon itself. This can be a global, national, regional, industry level or institutional.

Problem statement in general. Therefore, given the generally recognized importance of assessing the level of internationalization, the introduction of methods and tools for building reliable software systems for assessing the level of internationalization of scientific institutions is relevant and useful for both research and management applications. the development of a universal software design technology for automating the process of assessing the level of internationalization of the activities of a scientific institution will significantly simplify the process of developing software for the needs of managing the level of internationalization of scientific institutions of a specific community.

Analysis of recent research and publications. It is known [1; 2; 5; 6] that evaluating the level of internationalization of scientific institutions is a non-trivial task both due to the difficulty and complexity of the concept of internationalization itself, and due to its utilitarian nature, which leads to a multiplicity of possible utility functions (indicator systems) depending on the objective environmental conditions and its subjective vision by scientific communities.

At the same time, the analysis of the process of evaluating the level of internationalization makes it possible [9] to build its generalized model and identify functional requirements for a software system for its automation.

The analysis of the process of evaluating the level of internationalization highlights the fact that they are uncertain in advance and may change over time, in particular:

– the list and composition of data sources, formats and data access mode;
– methods:
- data normalization;
- data aggregation, indices and measurements;
- formal, in particular statistical, research of data and assessment results;
- ordering of scientific institutions;
- adoption of an expert decision on compliance with the established quality criteria;

– location, operating environment, presentation format, method of placement of materials intended for publication.

The aim of this paper is to determine the main stages of software design technology for automating the process of evaluating the level of internationalization of scientific institutions based on the proposed architectural template and software system models.

Presentation of the main material.

Structure of the architectural design template. Unlike universal design templates [3; 4; 7; 8; 10], which simplify the development of arbitrary software, the proposed architectural template [8; 11] IRI-Frame, the structure of which is presented in Figure 1, is aimed at solving the problem of choosing the architecture of a software system to automate the process of evaluating the level of internationalization of a scientific institution.

As can be seen from Figure 1, the IRI-Frame design template has the following structural components:

1. Expert is a class that provides user interaction with the IRIS class.

2. The IRIS class interacts simultaneously with the user, data provider, method provider, and publisher. It provides execution of all subprocesses of the internationalization level evaluation process by applying the

methods defined in it to the objects-properties of the DataProvider, MethodProvider, and PublicationMode types.

3. Data provider (DataProvider) is an interface for interacting with possible data sources. Its methods are available to the IRIS class and its descendants. The user of this system will be able to work with the data sources DataSource1, ProjectA and Specification25.

4. Concrete data providers (Concrete DataProvider) implements the DataProvider interface, providing interaction with individual concrete data sources.

5. Method provider (MethodProvider) is an interface that declares access to numerical and other computational methods. Its methods are available to the IRIS class and its descendants.

6. Concrete method providers (Concrete MethodProvider) implement the MethodProvider interface, providing access to concrete computational methods. Among the computational methods available to the user are the data normalization methods RankingNormalise, zScoreNormalise and MaxMinNormalise. The aggregation methods available are AdditiveAggregation, GeometricAggregation and MCAAggregation. The weighting methods are PCAWeighting. The statistical methods are CorrelationAnalysis.

7. Publisher (PublicationMode) is an interface that declares the publication of evaluation results. Its methods are available to the IRIS class and its descendants.

8. Concrete publishers (Concrete PublicationMode) implement the PublicationMode interface, providing publication in specific locations, operating environments, representation formats, etc. The user can publish the evaluation results in one of two modes – for viewing and with editing of test data (ViewMode and EditDataMode respectively).

Thus, the IRI-Frame architectural template provides work with different data sources, the use of various computational methods and the publication of results in different ways, and therefore allows you to use the same code base to create various system configurations. It should also be noted that the IRI-Frame architectural template can be implemented by combining universal GoF templates [1; 4], for example, the Bridge structural template to separate the abstraction of the index aggregation method from the aggregation algorithm, or the Adapter – eliminate the problem of processing heterogeneous data.

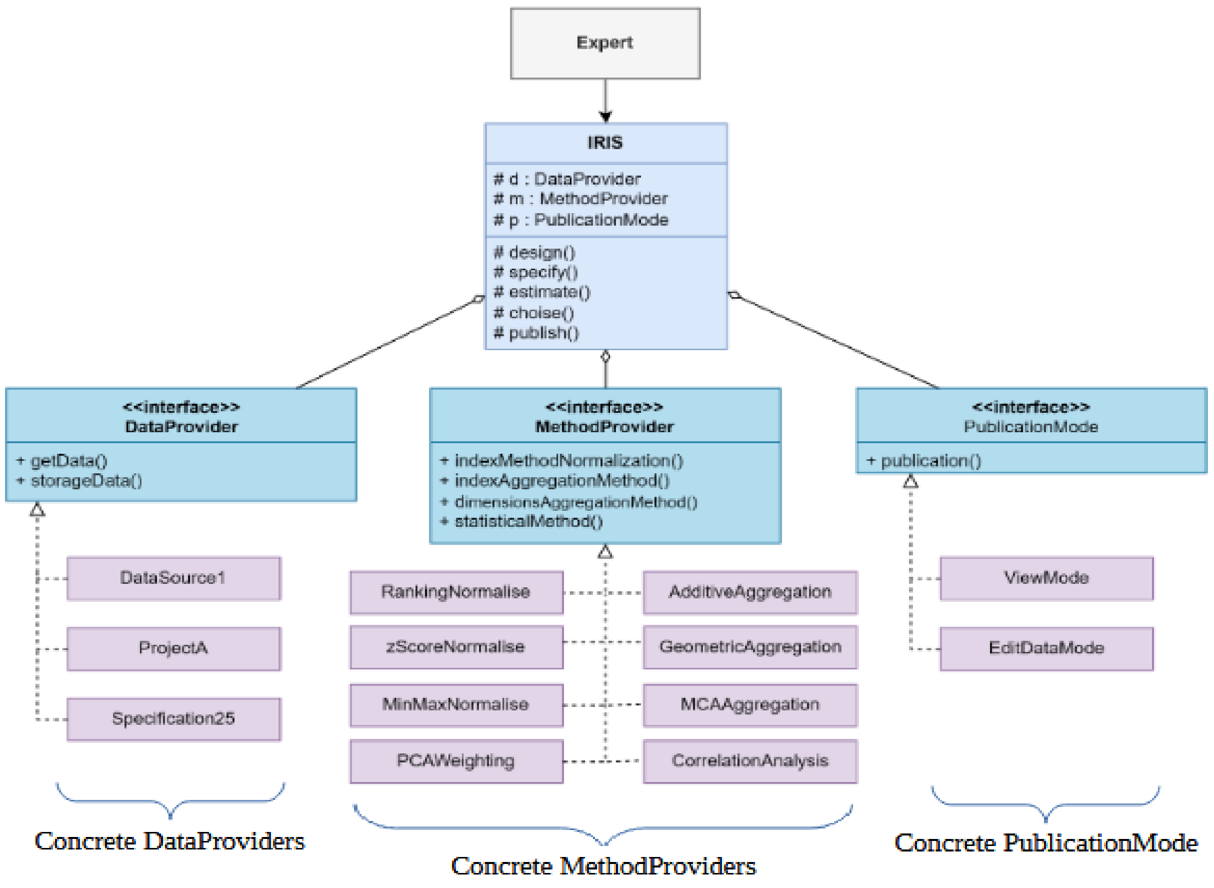

Fig. 1. Structure of the IRI-Frame design template

Software system models for automating the process of evaluating the level of internationalization. The processes and boundaries of the software system for evaluating the internationalization of scientific institutions [9] identified as a result of the analysis of the subject area allowed us to formulate requirements for it and propose a basic architecture with the separation of the structure at the level of modules and components [4].

The developed architectural design template allows us to separate the general functionality of the use cases from specific data, specific calculation and publication methods.

Comparing the structure of the basic architecture and the structure of the architectural design template, we come to the conclusion that it is necessary to further detail the representations of the software system.

Static model of a software system. The main operational functionality of the software system, as shown in (Fig. 1), in the architectural design pattern is concentrated in the IRIS class. However, given the list of use cases, such a concentration of functions is excessive, which requires decomposition of the IRIS class.

Figure 2 presents a class diagram as a static software model. The IRIS base class aggregates instances of classes-implementations of the DataProvider, MethodProvider and PublicationMode interfaces and provides the constructors conDesigner(), conSpecifier(), conEstimator(), conChoicer(), conPublisher() to create objects, respectively, of the Designer, Specifier, Estimator, Choicer and Publisher classes. These classes are descendants of the IRIS class and provide the functionality of the software system for automating processes:

– Designer – designing a system for evaluating the level of internationalization of a scientific institution;
– Specifier – researching the evaluation system;
– Choicer – choosing the methodology that best meets the quality requirements;
– Estimator – using the approved specification to assess the level of internationalization;
– Publisher – publishing the evaluation results.

Descendant classes inherit members of the ancestor class and may have additional ones.

Thus, the Designer class defines, in particular, methods for creating a Project object, saving it in the data warehouse and reading from the warehouse – createProject(): Project, saveProject() and getProject(): Project, respectively.

In the Project class, the object of which is aggregated by an instance of the Designer class, the methodology constructor createMethodology() and the list of created methodologies: List<Methodology> are defined.

The Specifier and Choicer classes use Project to access the defined methodologies in order to, respectively, specify them (Specifier) and select the one that best meets the quality criteria (Choicer).

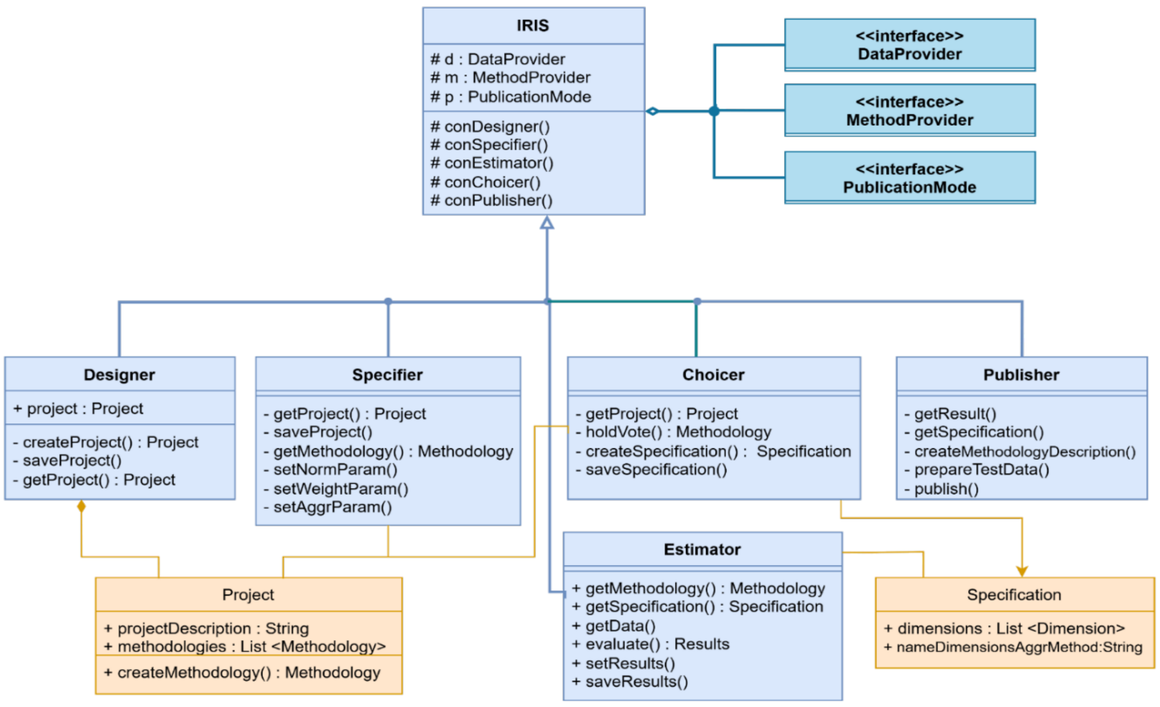

Fig. 2. Diagram of basic classes of the software system for evaluating the level of internationalization of a scientific institution

The methodology that was recognized as the best receives the status of a specification, becomes an independent artifact – Specification, the objects of which are stored in the data warehouse. The Specification class is structurally identical to the Methodology class. However, the Methodology instance, firstly, is not an independent artifact, but only a component of the Project composite, and secondly, it may be unspecified or partially specified. Instead, an instance of the Specification class is created as a clone of a fully specified methodology.

The classes-artifacts Project and Specification and their relationships with other classes are highlighted in beige on the diagram. The arrow on the association relationship from the Choicer class emphasizes the exclusive way to create a new instance of the Specification class.

The Specifier class contains, in particular, methods for reading and saving a possibly modified or newly defined project getProject() and saveProject(), methods for accessing project methodologies, in particular getMethodology(), and methods for setting parameters of normalization, weighting (weights) and aggregation methods – setNormParam(), setWeightParam() and setAggrParam(), respectively.

The Choicer class contains methods for reading the project getProject(), recognizing one of the studied methodologies as the best – holdVote(): Methodology, creating a specification and saving it in the data store – createSpecification(): Specification and saveSpecification() respectively.

The Estimator class defines methods for accessing the methodology, specification and data – getMethodology(), getSpecification() and getData() respectively, calculating the results evaluate(): Results and saving them setResults() in the methodology class or in the data store saveResults().

The Publisher class defines methods for accessing the results of the level evaluation getResult() and specification getSpecification(), as well as creating a methodology description, preparing a publication with test data and, in fact, publishing createMethodologyDescription(), prepareTestData() and publish().

Figure 3 presents the hierarchy of classes for representing Project and Specification artifacts. The diagram shows that the Project composite contains zero or more methodologies and a project description, which can be used to supplement the methodology description in an instance of the Specification class. The Methodology and Specification classes are identical in structure, as can be seen from the diagram.

The DataResult type depends on the structure of the internationalization level evaluation created during the operation of the software system, so it is not defined here. The same is true for the Parameters type, which depends on the chosen computational methods.

The Methodology and Specification classes contain a description of the methodology, a list of dimensions, names and parameters of the selected aggregation method for the dimensions.

The Dimension class contains the name and description of the dimension, a list of the measures that define it, and the method and parameters of the aggregation method for the measures.

The Index class contains the name and description of the measures, a list of paths to the data with the values of the measures, and the method and parameters of the normalization method for the measures.

Dynamic model of the software system for evaluating the level of internationalization of scientific institutions. To represent the temporal deployment of the functioning of the software system and the interaction of the selected classifiers, interaction diagrams are used, one of the varieties of which is a sequence diagram.

The diagram in (Fig. 4), presents a generalized software system sequence diagram. The diagram shows that, in the general case, the execution of the methods of each object is initiated by the system user. Thus, by contacting an object of the Designer class, the user can initiate the creation of a new project, and then proceed to the specification and study of each existing (created) methodology in it by contacting an instance of the Specifier class. After creating the project, the Designer and the Specifier, after specifying or studying the methodologies available in it, save the project in the data warehouse.

The Specifier specifies the methodologies created by the Designer and contacts the Estimator to perform calculations within the scenarios for determining the quality indicators of each of them.

An instance of the Choicer class, comparing the quality indicators of the specified project methodologies and, possibly, taking into account expert opinions on compliance with quality criteria, internationalization policy and strategy, chooses one of them, which is stored as a Specification. In the future, the Specification is used to calculate the level of internationalization of scientific institutions, until a decision is made to modify it or develop a new project.

Estimator, to determine the level of internationalization in accordance with the existing specification, loads the Specification, performs the evaluation and stores the results in the data warehouse, and references to them (with a time stamp) in the specification.

The Publisher class provides for loading the specification, and therefore the results of the internationalization level evaluation, generates a description of the evaluation methodology, and publishes the named materials.

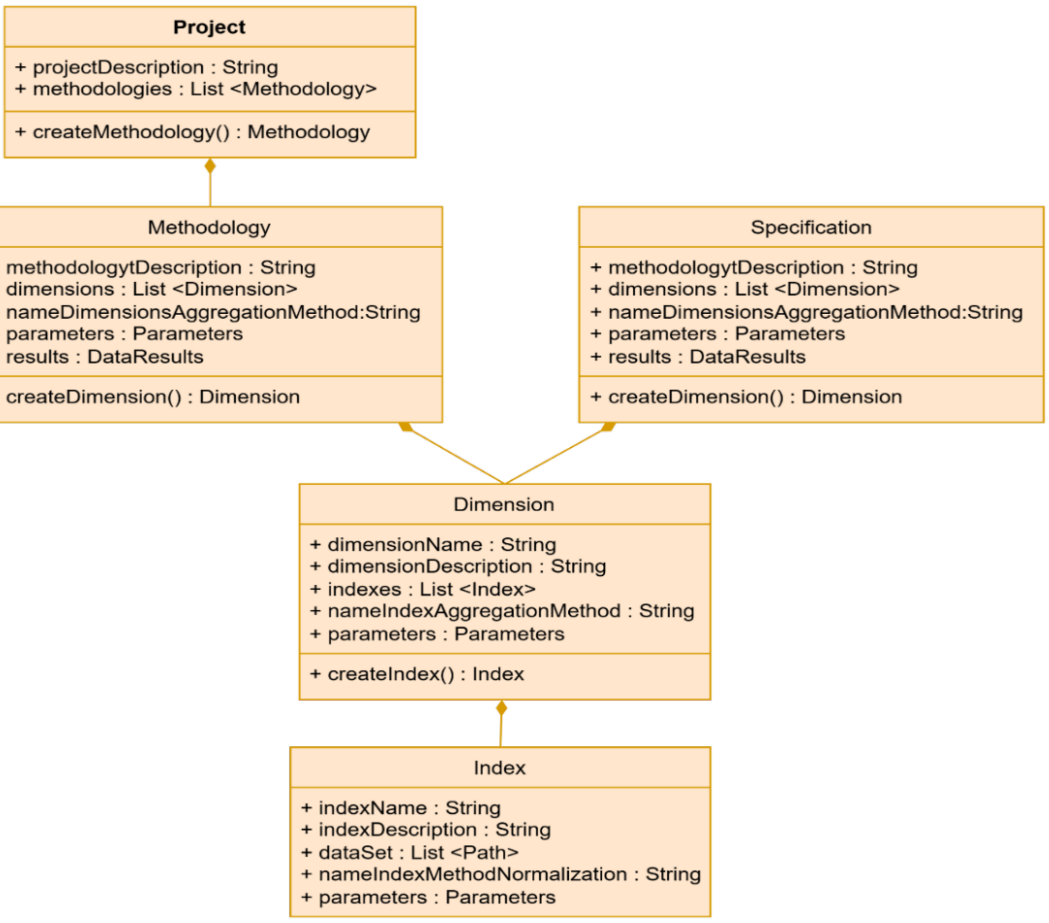
Project
+ projectDescription : String
+ methodologies : List <Methodology>
+ createMethodology() : Methodology
Methodology
+ methodologytDescription : String
+ dimensions : List <Dimension>
+ nameDimensionsAggregationMethod:String
+ parameters : Parameters
+ results : DataResults
+ createDimension() : Dimension
Specification
+ methodologytDescription : String
+ dimensions : List <Dimension>
+ nameDimensionsAggregationMethod:String
+ parameters : Parameters
+ results : DataResults
+ createDimension() : Dimension
Dimension
+ dimensionName : String
+ dimensionDescription : String
+ indexes : List <Index>
+ nameIndexAggregationMethod : String
+ parameters : Parameters
+ createIndex() : Index
Index
+ indexName : String
+ indexDescription : String
+ dataSet : List <Path>
+ nameIndexMethodNormalization : String
+ parameters : Parameters

Fig. 3. Structure of the main artifact classes

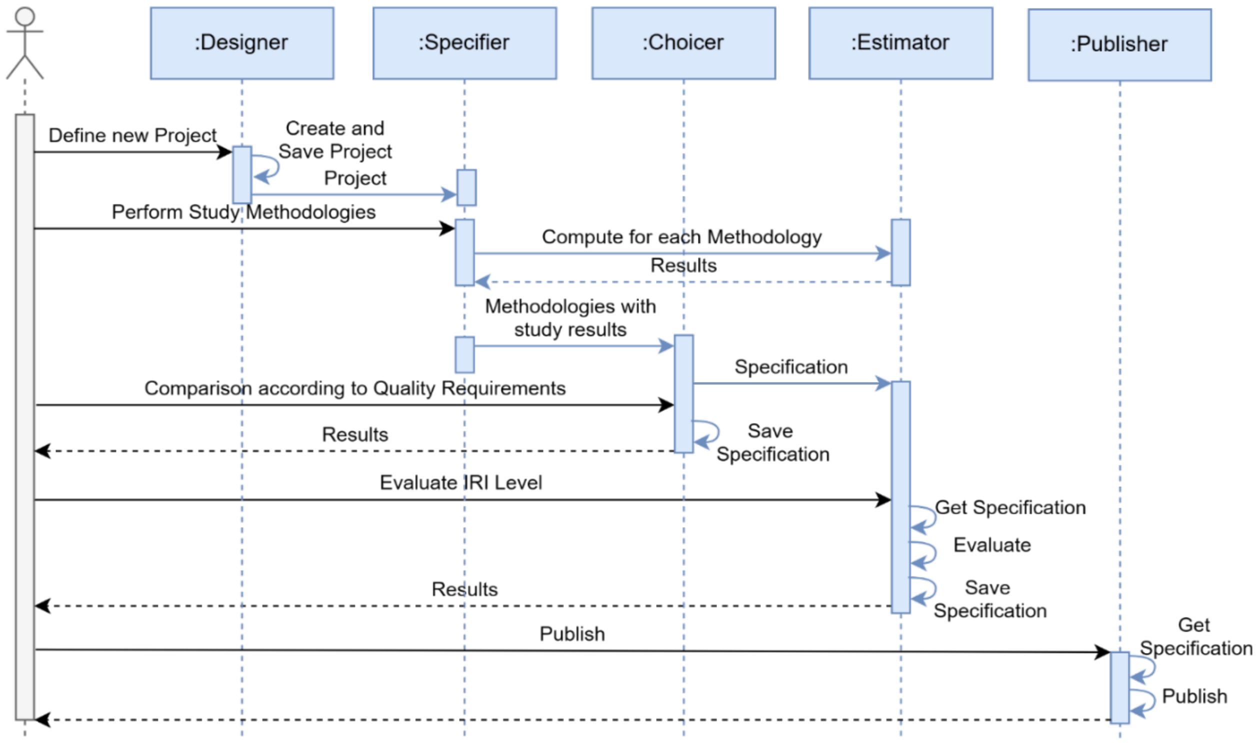

Fig. 4. Software system sequence diagram

Technology for designing and modifying software tools to automate the process of evaluating the level of internationalization. Software tools for automating the process of evaluating the level of internationalization of scientific institutions can be implemented both in the form of a separate system and in the form of a component of another system.

The life cycle of a software system, as is known, involves iterative processes of its development and modification, but does not exclude the identification of certain stages, each of which, in the general case, can be performed repeatedly.

Let us consider the main (enlarged) stages of designing a separate software system for automating the process of evaluating the level of internationalization of scientific institutions, provided that there is a formal or informal interested scientific community.

The first stage involves determining the organizational and operational environment of the software system, in particular:

– delineation of the boundaries of the system and its surroundings, as well as their (potential) representatives – experts (Experts), managers (IManager), public users and, possibly, other categories of actors;

– determination of the place of the software system in the system of activity of actors and the scientific community.

Determining these aspects allows us to formulate requirements for the system for evaluating the level of internationalization of scientific institutions in order to adapt the basic architecture proposed here to the needs of users.

The second stage involves identifying high-level representations of the substantive component of the system's functionality, in particular:

– the presence or formulation of documented interpretations of the concepts of internationalization of scientific institutions and the level of internationalization. Such documents are usually the Internationalization Policy and/or the Internationalization Strategy;

– the presence of a documented list and interpretations of the quality criteria for the system for evaluating the level of internationalization.

The specified high-level positions allow to define certain requirements for the software system and its information component:

– requirements for the necessary source information on the activities of scientific institutions. This allows to draw a conclusion about the possibility of using existing, possibly external, sources or the need to develop a separate subsystem for collecting relevant data;

– methods of evaluating the level of internationalization of scientific institutions. This allows to limit the set of methods of normalization, aggregation and ordering of data, as well as the necessary set of statistical methods.

– requirements for methods of presenting the results of the evaluation and description of the methodology, evaluation. This allows to limit the set of methods for presenting materials to be published.

The third stage involves designing the components of the software system:

– universal components of the software system that provide support for the processes of creating a system for evaluating the level of internationalization of scientific institutions and performing such an evaluation;

– special components that implement such computational methods, such as: methods of normalization and aggregation of data, weighting methods, ordering methods, statistical methods, etc. Special components should also include components that provide data access mechanisms and publication mechanisms.

When developing a software system to automate the process of evaluating the level of internationalization of scientific institutions, it is advisable to use the IRI-Frame architectural design template proposed in this study.

Conclusions. The results presented in this article on the research process of designing software for automating activities aimed at evaluating the level of internationalization of scientific institutions allow us to formulate the following conclusions:

1. The architectural design template has been developed that provides isolation of the code of universal components that provide support for the processes of creating a system for evaluating the level of internationalization from special components that implement computational methods, data access methods and methods related to the publication of materials.

2. The static model of a software system has been developed using the developed architectural design template, which contains behavioral classes, abstract classes (interfaces) and essential classes for the main artifacts of the software system.

3. The dynamic model of interaction of software entities of the system for the main use cases of the system has been developed.

4. The main stages of the technology of designing software tools for automating the process of evaluating the level of internationalization of scientific institutions are proposed.

5. All the results presented in this section are aimed at simplifying the process of developing software for the needs of managing the level of internationalization of scientific institutions of a specific community.

Bibliography:

1. Bas M. C., Boquera M., Carot J. M. Measuring internationalization performance of higher education institutions through composite indicators, INTED, 2017 Proceedings, 2017. pp. 3149–3156. DOI: 10.21125/inted.2017.0815.

2. Brandenburg U., and G. Federkeil. "How to Measure Internationality and Internationalisation of Higher Education Institutions. Indicators and Key Figures." 2007. (Accessed February 09, 2024). URL: https://www.che.de/en/download/how_to_measure_internationality_ap_92-pdf (open in a new window).

3. Freeman E., Robson E. Head first design patterns. O'Reilly Media, 2020, p. 669.

4. Gamma E., Helm R., Johnson R., and Vlissides J. Design Patterns. Reading, MA.: Addison-Wesley, 1995.

5. Knight J. Monitoring the Quality and Progress of Internationalization. *Journal of Studies in International Education*, 2001. 5(3), 228–243. DOI: 10.1177/102831530153004.

6. Knight Jane. Internationalization Remodeled: Definition, Approaches, and Rationales. *Journal of Studies in International Education.* 2004. 8. 5–31. DOI: 10.1177/1028315303260832.

7. Larman C. Applying UML and patterns: an introduction to object-oriented analysis and design and iterative development. Pearson Education India, 2012. p. 703.

8. Mayvan B. B., Rasoolzadegan A., Yazdi Z. G. The state of the art on design patterns: A systematic mapping of the literature. *Journal of Systems and Software*, 2017. Vol. 125, P. 93–118.

9. Statyvka Y. I., Nedashkivskiy O. L., Mingjun Z. Model of the process for evaluating the level of internationalization of the scientific institution activities. *Connectivity,* No. 3 (175), 2025, pp. 42–51. DOI: 10.31673/2412-9070.2025.020915.

10. Wedyan F., Abufakher S. Impact of design patterns on software quality: a systematic literature review. *IET Software*, 2019. Vol. 14, Issue 1, P.1–17. DOI: 10.1049/iet-sen.2018.5446.

11. Дичка І. А., Сулема О. К., Крайносвіт А. А. Програмна система логістичного обліку на основі дворівневого штрихового коду. Системні технології*,* 2020. № 6, С. 28–38.

Дата надходження статті: 18.09.2025
Дата прийняття статті: 20.10.2025
Опубліковано: 04.12.2025