BULLETIN of Cherkasy State Technological University

Journal homepage: https://bulletin-chstu.com.ua/en

Vol. 29 No. 3. 2024

UDC 004.415.2:378
DOI: 10.62660/bcstu/3.2024.42

Using design patterns and typed languages in the development of an adaptive model of personalised learning

Pavlo Fedorka*

Doctor of Philosophy, Associate Professor
Uzhhorod National University
88000, 3 Narodna Sq., Uzhhorod, Ukraine
https://orcid.org/0000-0002-9242-5588

Fedir Saibert

Master, Assistant
Uzhhorod National University
88000, 3 Narodna Sq., Uzhhorod, Ukraine
https://orcid.org/0009-0004-8081-4174

Roman Buchuk

PhD in Physical and Mathematical Sciences, Associate Professor
Uzhhorod National University
88000, 3 Narodna Sq., Uzhhorod, Ukraine
https://orcid.org/0009-0000-4199-5583

Abstract. The purpose of this study was to determine the effectiveness of using design patterns and typed programming languages, specifically TypeScript and C#, in building an adaptive model of personalised learning in software engineering. The study examined the use of design patterns in the development of an adaptive model of personalised learning, reviewed the use of TypeScript and C# in the creation of such a model, and compared these typed programming languages and resources for software engineering education. The key findings of the study showed that among the design patterns, Singleton, Factory, Strategy, and Observer are the most effective for building an adaptive personalised learning model, as they increase the flexibility and adaptability of the system. The developed software prototypes showed that the use of the TypeScript language ensures the reliability of the adaptive system due to static typing and flexible interfaces, while the C# language with Generics and Language Integrated Query (LINQ) capabilities contributes to effective data management and modular integration. The comparative analysis revealed that C# is better suited for more complex systems with higher data management requirements, while TypeScript provides fast integration and greater flexibility in front-end development. A review of the available learning resources for both languages also revealed a greater variety for TypeScript, which may facilitate faster learning for new users. The conclusions showed that the use of design patterns and typed programming languages is an essential approach to creating personalised learning models that can adapt to individual user needs and increase the effectiveness of software engineering education

Keywords: individualised education; differentiated learning; software architecture; object-oriented modelling; customisable interfaces; data optimisation

Article's History: Received 17.06.2024; Revised 10.09.2024; Accepted 21.10.2024

Suggested Citation:
Fedorka, P., Saibert, F., & Buchuk, R. (2024). Using design patterns and typed languages in the development of an adaptive model of personalised learning. *Bulletin of Cherkasy State Technological University*, 29(3), 42-54. doi: 10.62660/bcstu/3.2024.42.

*Corresponding author

INTRODUCTION

A modern approach to personalised learning involves adapting educational content to meet the individual needs of users, making it a valuable component of educational technology. Developing adaptive systems requires not only a flexible architecture but also the use of programming languages that support reliability and scalability. For example, TypeScript allows implementing strict type control in web applications, which increases their reliability and scalability, especially in large projects. C#, on the other hand, is a high-level programming language that has powerful data processing tools that enable efficient data management and optimisation of complex systems. The combination of these features makes both languages suitable for developing adaptive learning systems that can integrate and analyse learning content according to the needs of users.

Overall, adaptive learning systems face a series of challenges that limit their effectiveness and flexibility in personalising content. Specifically, the lack of standardised approaches to implementing the architecture of such systems leads to major difficulties in ensuring their scalability and modularity. The integration of structured and efficient development approaches, such as design patterns, is necessary to ensure greater stability and adaptability of the system to changing user needs. However, many existing approaches focus on general architectural solutions, without paying due attention to concrete programming languages and the ways in which their capabilities can affect the quality of adaptive systems.

For example, N. Pravorska & S. Hryha (2024) focused on the value of design patterns for effective microservice architecture. The researchers emphasised the importance of such patterns as Circuit Breaker, API (Application Programming Interface) Gateway, and Saga, noting that they allow preserving the integrity of the system in case of microservice failures. Q. Gou & H. Poliakova (2024) examined personalised learning in the digital learning environment, focusing on building an adaptive model. They emphasised the significance of systematicity, adaptability, and individual approach, which are key to the development of personalised learning models using digital resources. Furthermore, O. Koshova *et al.* (2023) pointed out the advantages of using TypeScript to develop a distance learning system, noting its effectiveness in combination with the React library to create an interactive learning environment.

M. Mirzaei & K. Meshgi (2023) showed how machine learning-based systems can provide personalised learning by adapting to the level of knowledge, interests, and needs of students. The developed Partial and Synchronised Caption model uses automatic analysis and processing of educational content, which allows the system to adapt learning materials to each user. P. Peng & W. Fu (2022) demonstrated methods for recognising patterns of personalised adaptive learning in online education. Based on the analysis of learning behaviour, clustering, and correlation analysis, a characteristic model was built to adapt learning resources to the level of knowledge, learning style, interactive behaviour, and social learning of users. C. Halkiopoulos & E. Gkintoni (2024) showed that artificial intelligence (AI) contributes to the improvement of personalised learning and adaptive assessment and recommended the development of algorithms to reduce bias and further research into the ethical aspects of AI in education. T. Ball *et al.* (2019) presented Static TypeScript, which is a subset of TypeScript and is designed to teach programming on microcontrollers. The researchers pointed out that Static TypeScript compiles to machine code in a browser, making TypeScript an effective solution for educational projects on small devices.

Moreover, C. Troussas *et al.* (2020) investigated adaptive programming learning using C#, applying the Revised Bloom Taxonomy learning theory. The findings showed that the used approach considerably improves student performance compared to systems without adaptability. L. Chai *et al.* (2024) presented the Adaptive Information Fusion personalised learning algorithm, which helps to increase the model's adaptability on local data after each training cycle. The findings confirmed the effectiveness of Adaptive Information Fusion in personalised learning systems with heterogeneous data. C.T. Dumitru's (2024) study pointed to the potential of adaptive learning systems in the context of higher education, emphasising their role in personalising education through AI and data analytics. The findings showed that such systems can greatly improve learning outcomes and increase student satisfaction.

These studies focused on adaptive models of personalised learning rather than design patterns and typed programming languages, as is the case in the present study. Overall, the purpose of this study was to determine the effectiveness of using design patterns and typed programming languages to build an adaptive learning system. The objectives included analysing the use of design patterns to develop an adaptive personalised learning model, reviewing the TypeScript and C# programming languages in the context of creating this model, and comparing these languages and learning resources for software engineers.

MATERIALS AND METHODS

The study was divided into three stages, which included the analysis of an adaptive personalised learning model based on design patterns and typed programming languages TypeScript and C#. At the initial stage, a thorough analysis of various design patterns was performed to determine the most suitable ones for building an adaptive personalised learning model. The patterns considered were Factory, Abstract Factory, Builder, Prototype, Singleton, Adapter, Bridge, Composite, Decorator, Facade, Flyweight, Proxy, Chain of Responsibility, Command, Iterator, Mediator, Memento, Observer, State, Strategy, Template Method, and Visitor. Each pattern was analysed for its ability to improve the flexibility, adaptability, and scalability of the system. Among them, the most suitable patterns for developing an adaptive learning model were selected. On their basis, an architectural diagram of the model was built, which showed examples of scenarios for the use of each pattern. Furthermore, the relationships between Factory, Strategy, Observer, and Singleton patterns were considered.

The second stage was to apply the typed programming languages TypeScript and C# to implement the adaptive elements of the learning model. The features of the TypeScript language were reviewed, including its static typing capabilities, interfaces, and support for the object-oriented approach. Based on this, a simple software prototype was created to demonstrate the use of types and interfaces to improve system reliability. This programme was written using the TS Playground online compiler. The TypeScript program defined the LearningModule and User interfaces, the LearningSystem class, and the getRecommendedModule method, which provided module recommendations based on the user's profile. On the other hand, the adaptive system application was written in C#, using Generics and Language Integrated Query (LINQ) capabilities to optimise data management and facilitate access to modular components. For this, the Online C# Compiler platform was used. This programme implemented the ILearningModule interface and the LearningModule class, as well as the LearningSystem class with the AddModule method, which helped to dynamically add new modules to the system, maintaining its adaptability.

At the final stage of the study, a comparative analysis of the TypeScript and C# programming languages was performed in the context of their use for the development of adaptive educational models. This analysis included a comparison of the approaches to typing in both programming languages, including strong and weak typing, as well as static and dynamic type checking. To demonstrate the capabilities of each language, examples of using the Strategy and Singleton patterns to adapt the training content were implemented. The TypeScript example, which used the Strategy pattern, included the AdaptationStrategy interface and the BasicAdaptation and AdvancedAdaptation classes, which allowed creating various adaptation strategies and choosing the right one for each user. An adaptive system was implemented in C# using the Singleton pattern for managing user data, which provided a single access to user profile information in the system (Fenton, 2018; Skeet, 2019). The study also compared the learning resources available for both languages, such as documentation, online courses, books, video tutorials, Integrated Development Environment (IDE), frameworks, testing tools, and package managers.

RESULTS

Application of design patterns in the development of an adaptive model of personalised learning. Design patterns are proven solutions to common problems that arise in the software development process. Their use allows creating code that is more flexible, extensible, and easy to maintain. In the context of adaptive learning systems, design patterns help to structure complex systems, reducing their overall complexity, and increase code repeatability by enabling the use of ready-made solutions for common problems. These solutions also improve code comprehension for other developers, as familiar patterns make it easier to understand. Furthermore, patterns make the system more flexible, allowing for easy changes and additions. The choice of particular patterns for the adaptive learning model is based on their functionality and ability to meet the system's requirements (Table 1).

Table 1. Design patterns description

Design pattern	Description
Factory	Creates objects through a method, allowing subclasses to determine which class to set
Abstract Factory	Provides an interface for creating a group of related or dependent objects without specifying their particular classes
Builder	Separates the construction of a complex object from its representation, which allows creating multiple representations of the object
Prototype	Allows creating new objects by cloning existing prototype objects
Singleton	Provides the creation of only one instance of a class with a global access point to it
Adapter	Converts a class interface to another interface expected by the client, ensuring compatibility between incompatible interfaces
Bridge	Separates abstraction and implementation so that they can change independently of each other
Composite	Allows processing individual objects and their compositions in the same way, creating a hierarchy of objects
Decorator	Dynamically adds new functionality to an object without changing its structure
Facade	Provides a simplified interface to more interconnected classes or subsystems
Flyweight	Optimises memory usage by sharing a common state between many small objects
Proxy	Provides a surrogate or substitute for another object, controlling access to it
Chain of Responsibility	Passes the request to a sequence of objects, where each object decides whether to process the request or pass it on

Continued Table 1.

Design pattern	Description
Command	Encapsulates a request as an object, enabling the parameterisation of clients with different requests and providing the ability to cancel operations
Iterator	Provides consistent access to collection items without revealing its internal structure
Mediator	Defines an object that encapsulates the way objects interact, reducing dependencies between them
Memento	Preserves and restores the previous state of the object without breaking its encapsulation
Observer	Provides a mechanism for one object to notify others of changes in its state
State	Allows an object to change its behaviour when its state changes, as if the object were changing its class
Strategy	Defines a family of algorithms, encapsulates them, and allows changing the algorithm independently of the client using it
Template Method	Defines the skeleton of the algorithm in a superclass, allowing subclasses to define some steps of the algorithm
Visitor	Adds new operations to classes without changing their structure by moving the operation logic to a separate object

Source: created by the authors

Incorporating design patterns into an adaptive learning model not only facilitates the development process but also ensures high quality and reliability of learning solutions, which is vital for success in the modern educational sphere. To develop an adaptive personalised learning model, the most effective patterns are Factory, Strategy, Observer, and Singleton. For example, the Singleton pattern provides class uniqueness and global access to it, which is useful for presenting a single user context where all information about their progress, preferences, and settings is stored. The Factory pattern allows defining an interface for creating objects, and subclasses can decide which particular instance should be created. This is extremely convenient for creating various types of learning materials (videos, text assignments, tests) according to the user's needs. The Strategy pattern, in turn, allows forming a family of algorithms, encapsulating each of them and making them interchangeable. In the case of an adaptive learning system, this pattern can be used to implement various adaptation algorithms, such as recommendation systems or content branching algorithms. The Observer pattern provides a one-to-many dependency that allows automatically updating all objects that depend on one when it changes. This can be useful for updating the user interface when their data or learning outcomes change. Example scenarios for using these patterns clearly demonstrate their expediency (Fig. 1).

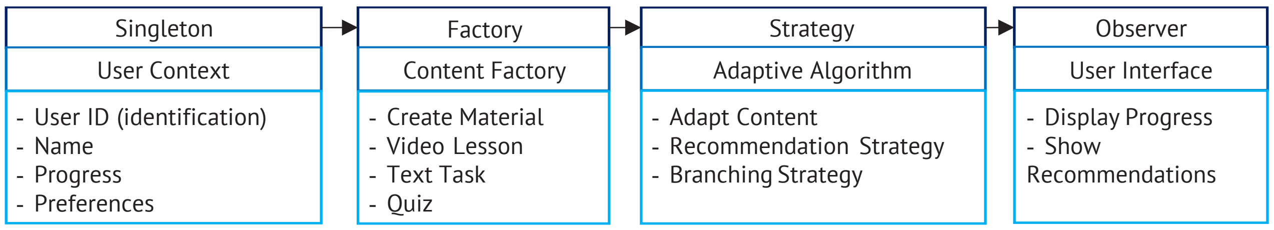


Figure 1. Diagram of the relationships between different components of the system and their functions in the context of adaptive learning
Source: created by the authors

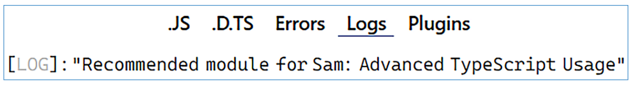
The architecture diagram of the adaptive learning model illustrates how these patterns function together to optimise the learning process. Notably, the use of design patterns considerably improves the architecture of an adaptive learning model, making it more flexible and adaptive to the needs of users. When implementing these patterns, one should consider their specific features in typing and structure. For example, TypeScript, due to its static typing and flexible type management capabilities, is an effective tool for creating reliable and adaptive code that facilitates dynamic adaptation of learning content. C#, on the other hand, provides more rigorous typing and a powerful infrastructure to support complex structures and functional relationships in the system, which ensures a great level of reliability but requires more precise adherence to syntactic and type constructs.

Using typed languages TypeScript and C# to create an adaptive model of personalised learning. TypeScript is a powerful JavaScript-based programming language that provides static typing. This means that developers can define data types for variables, functions, and objects, which helps to prevent many types of errors at the compilation stage. The key feature of TypeScript is its static typing, which allows programmers to clearly define data types. This results in fewer errors, improved readability, better integration, and enhanced functionality. That is, static typing allows detecting errors at the early stages of development, even before the code is executed. Explicit data types make the code easier to read, as other developers can quickly understand what data types are used. Furthermore, TypeScript integrates well with existing JavaScript libraries, making it easy to use existing solutions. This language supports modern JavaScript features, including asynchronous functions and result handling, making it suitable for creating complex asynchronous systems. These features make TypeScript a reliable choice for developing educational systems where it is important to maintain strict code quality and ensure reliability during the execution of learning algorithms.

When developing an adaptive learning model in TypeScript, it is essential to properly organise the code structure and manage dependencies. Using interfaces and classes, a clear definition of data types and their interaction can be ensured. Below is a code sample to illustrate the management of types and interfaces in TypeScript:

```
// Define an interface for a learning module
interface LearningModule {
 title: string;
 content: string;
 level: number; // Difficulty level
}

// Define an interface for a user
interface User {
 id: number;
 name: string;
 progress: number; // Percentage of course
completion
}

// Class for the learning system
class LearningSystem {
 private modules: LearningModule[] = [];
 private user: User;

 constructor(user: User) {
 this.user = user;
 }

 // Method to add a module to the system
 addModule(module: LearningModule) {
 this.modules.push(module);
 }

 // Method to get a recommended module for
the user
  getRecommendedModule(): LearningModule |
null {
   const suitableModules = this.modules.
filter(module => module.level <= this.user.
progress);
   return suitableModules.length > 0 ?
suitableModules[suitableModules.length - 1]
: null;
 }
}

// Creating a user object
const user: User = {
 id: 1,
 name: "Sam",
 progress: 50 // User has completed 50% of
the course
};

// Creating the learning system
const learningSystem = new
LearningSystem(user);

// Adding modules to the system
learningSystem.addModule({ title:
"Introduction to TypeScript", content:
"Learning about types", level: 25 });
learningSystem.addModule({ title: "Advanced
TypeScript Usage", content: "Diving into
interfaces", level: 50 });
learningSystem.addModule({ title: "Modules
in TypeScript", content: "Structuring code",
level: 75 });

// Getting the recommended module
const recommendedModule = learningSystem.
getRecommendedModule();
if (recommendedModule) {
 console.log(`Recommended module for ${user.
name}: ${recommendedModule.title}`);
} else {
  console.log("No available modules for
recommendation.");
}
```

This programme demonstrates the use of TypeScript to create a simple adaptive learning system that recommends learning modules based on the user's progress. It illustrates the benefits of typing in TypeScript to ensure code is robust and readable. First, two interfaces are defined: LearningModule, which describes the learning module, and User, which represents the user and their data. Next, a LearningSystem class is created that manages the modules and users, with methods for adding modules and receiving recommendations based on progress. When the programme starts, a user with a certain progress is created, and modules of different difficulty levels are added. When the recommendation method is called, the system filters the modules, selecting those that match or are lower than the user's progress level. The result of the programme is to display the title of the recommended module for the user (Fig. 2).



```
[LOG]: "Recommended module for Sam: Advanced TypeScript Usage"
```

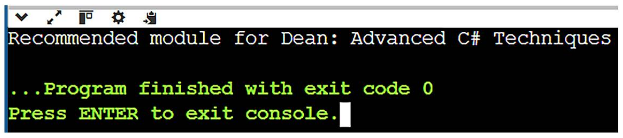
Figure 2. Result of the programme in TypeScript
Source: created by the authors

Thus, the code demonstrates how typed programming languages can be used to create adaptive learning systems that improve the quality of the learning process by customising the learning content. After reviewing the capabilities of TypeScript in creating adaptive learning systems, it is worth looking at C# as another powerful typed programming language that has its specific features and benefits.

C# is a powerful programming language that provides typing capabilities for building robust and interactive responsive learning systems. With features such as Generics and LINQ, C# allows developers to efficiently manage data and implement algorithms that help personalise learning. When developing an adaptive learning model in C#, it is crucial to provide a clear programme architecture that will enable convenient management of various components of the learning modules. For example, the use of Generics ensures type safety when working with multiple data types, while LINQ facilitates querying collections, which greatly simplifies the processing of information about training modules and users. Below is a code sample that demonstrates the implementation of an adaptive learning system in C#:

```
using System;
using System.Collections.Generic;
using System.Linq;

// Define an interface for a learning module
public interface ILearningModule
{
 string Title { get; }
 string Content { get; }
 int Level { get; } // Difficulty level
}

// Define a class for a learning module
public class LearningModule : ILearningModule
{
 public string Title { get; private set; }
 public string Content { get; private set; }
 public int Level { get; private set; }

 public LearningModule(string title, string
content, int level)
 {
 Title = title;
 Content = content;
 Level = level;
 }
}

// Define a class for a user
public class User
{
 public int Id { get; }
 public string Name { get; }
 public int Progress { get; } // Percentage
of course completion

  public User(int id, string name, int
progress)
 {
 Id = id;
 Name = name;
 Progress = progress;
 }
}

// Class for the learning system
public class LearningSystem
{
 private List<ILearningModule> modules = new
List<ILearningModule>();
 private User user;

 public LearningSystem(User user)
 {
 this.user = user;
 }

 // Method to add a module to the system
  public void AddModule(ILearningModule
module)
 {
 modules.Add(module);
 }

 // Method to get a recommended module for
the user
public ILearningModule GetRecommendedModule()
 {
 return modules.Where(module => module.Level
<= user.Progress).OrderByDescending(module
=> module.Level).FirstOrDefault();
 }
}

// Example usage
public class Program
{
 public static void Main()
 {
 // Creating a user object
 var user = new User(1, "Dean", 50); // User
has completed 50% of the course

 // Creating the learning system
 var learningSystem = new LearningSystem(user);

 // Adding modules to the system
               learningSystem.AddModule(new
LearningModule("Introduction to C#",
"Learning the basics", 25));
               learningSystem.AddModule(new
LearningModule("Advanced C# Techniques",
"Deep dive into generics", 50));
               learningSystem.AddModule(new
LearningModule("Design Patterns in C#",
"Understanding common patterns", 75));

 // Getting the recommended module
  var recommendedModule = learningSystem.
GetRecommendedModule();
 if (recommendedModule != null)
 {
 Console.WriteLine($"Recommended module for
{user.Name}: {recommendedModule.Title}");
 }
 else
 {
 Console.WriteLine("No available modules for
recommendation.");
 }
 }
}
```

This example shows how typing can be used in C# to create interfaces for training modules and users, as well as implement methods for making recommendations based on user progress. First, the interfaces for the training modules and the user class are defined. Then, a learning system class is created that manages the modules and users. After running the programme, the system recommends modules that are at or below the user's progress level. The result of the programme displays the title of the recommended module for the user, which emphasises the personalisation of learning in C# (Fig. 3).



```
Recommended module for Dean: Advanced C# Techniques

...Program finished with exit code 0
Press ENTER to exit console.
```

Figure 3. Result of the programme in C#
Source: created by the authors

Thus, C# provides effective tools for developing adaptive systems that improve the quality of the learning process by customising content according to the needs of users. Notably, the use of design patterns and typed languages TypeScript and C# in the development of an adaptive model of personalised learning shows that the integration of different patterns, such as Factory, Strategy, Observer, and Singleton, ensures high flexibility and scalability of the system, enabling the educational content to be quickly tailored to the individual needs of the user.

Comparative analysis of TypeScript and C# programming languages and training resources for software engineers. It is worth noting that TypeScript and C#, albeit typed languages, use distinct models to ensure strict control over the code (Table 2). This leads to differences in flexibility, reliability, and the amount of code that developers can easily adapt to changing requirements.

Table 2. Comparison of approaches to typing and structuring code between TypeScript and C#

Aspect	TypeScript	C#
Typing	Strong, static typing, support for structural typing, which allows flexibility in interfaces	Strong, static typing, nominal typing ensures high type rigour
Static type validation	Executed when compiling, allows quick error detection	Strict static checking during compilation, which improves reliability
Flexibility	High, with the ability to ignore some typing rules	Lower flexibility, focused on strict type matching
Structuring model	Simple modular approach, allows easy integration with web technologies	More sophisticated object-oriented model with properties, events, delegates
Interface support	Supports interfaces and modules for code decomposition	Support for interfaces and rich class model for deep hierarchy
Usage environment	Mostly used in web development	Used in both web and desktop applications with strong support for OOP (object-oriented programming)

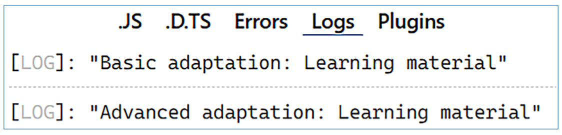
Source: created by the authors

That is, TypeScript typing is strong and static, which allows detecting errors during compilation, while C# typing is also static, but has greater strictness in type adherence during development. In TypeScript, more flexibility is possible due to structural typing, while C# is focused on nominal typing, which increases code reliability but reduces flexibility in its adaptation.

Design patterns continue to be crucial tools for ensuring the efficient structure and functionality of software systems. When developing an adaptive personalised learning model, the choice between TypeScript and C# becomes especially relevant, as both languages have their specific features in the implementation of such patterns as Singleton, Factory, Strategy, Observer, etc. Therefore, it is essential to consider examples of implementing these patterns in TypeScript and C#. For instance, the Strategy pattern in TypeScript is well suited for an adaptive system because it enables an easy change of algorithms to adapt the learning content depending on the user's needs. TypeScript with its structural typing provides a convenient way to implement strategies for multiple types of adaptation. Below is an example of implementing the Strategy pattern in TypeScript to adapt learning content:

```
// Interface defining the adaptation strategy
interface AdaptationStrategy {
 adapt(content: string): string;
}

// Basic adaptation class
class BasicAdaptation implements
AdaptationStrategy {
 adapt(content: string): string {
 return `Basic adaptation: ${content}`;
  }
}

// Advanced adaptation class
class AdvancedAdaptation implements
AdaptationStrategy {
 adapt(content: string): string {
 return `Advanced adaptation: ${content}`;
  }
}

// LearningSystem class allowing for flexible
adaptation strategies
class LearningSystem {
 private strategy: AdaptationStrategy;

  constructor(strategy: AdaptationStrategy)
{
 this.strategy = strategy;
 }

 adaptContent(content: string): string {
 return this.strategy.adapt(content);
 }

  setStrategy(strategy:  AdaptationStrategy)
{
 this.strategy = strategy;
 }
}

// Using the Strategy pattern:
const   system   =   new   LearningSystem(new
BasicAdaptation());
console.log(system.adaptContent("Learning
material")); // Basic adaptation: Learning
material
system.setStrategy (new
AdvancedAdaptation());
console.log(system.adaptContent("Learning
material"));    //    Advanced    adaptation:
Learning material
```

In this example, the Strategy pattern allows changing the adaptation strategy depending on the conditions, e.g., the student's level of training. This makes the system more flexible and dynamic, which is important for personalised learning. As a result, the application displays messages that demonstrate a variety of strategies for adapting learning content (Fig. 4).



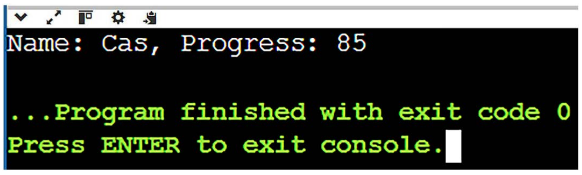
```
[LOG]: "Basic adaptation: Learning material"
[LOG]: "Advanced adaptation: Learning material"
```

Figure 4. Result of the TypeScript programme for the Strategy pattern
Source: created by the authors

In turn, the Singleton pattern is well suited for C# in the context of an adaptive learning model, as it allows having only one instance of a certain object in the system, e.g., storing current user data or system settings. This is especially convenient in C#, where Singleton is implemented with extra protection against multithreading. Below is an example of implementing the Singleton pattern in C# to manage the user context in an adaptive learning system:

```
using System;

public class UserData {
 public string Name { get; set; }
 public int Progress { get; set; }
}

public class UserContext {
 private static UserContext instance;
  private  static  readonly  object  lockObj  =
new object();
 private UserData userData;

 private UserContext() {}

 public static UserContext Instance {
 get {
 lock (lockObj) {
 if (instance == null) {
 instance = new UserContext();
 }
 return instance;
 }
 }
 }

 public void SetUserData(UserData data) {
 userData = data;
 }

 public UserData GetUserData() {
 return userData;
 }
}

class Program {
 static void Main() {
 // Using Singleton:
        UserContext.Instance.SetUserData(new
UserData { Name = "Cas", Progress = 85 });
 UserContext user = UserContext.Instance;

 // Output user data
      Console.WriteLine($"Name:      {user.
GetUserData().Name},    Progress:     {user.
GetUserData().Progress}");  //   Name:   Cas,
Progress: 85
 }
}
```

In this example, the programme, using the Singleton pattern for the UserContext class, allows storing user data, which includes the name and progress. In the Main method, the user's data with their name and progress is set using SetUserData. Next, both the user's name and progress are displayed, which demonstrates that the programme provides access to all user information through a single UserContext object (Fig. 5).



```
Name: Cas, Progress: 85

...Program finished with exit code 0
Press ENTER to exit console.
```

Figure 5. Result of the C# programme for the Singleton pattern
Source: created by the authors

Apart from reviewing the TypeScript and C# pattern implementations, attention should be paid to the learning resources and support for developing adaptive learning solutions in these languages. Such resources play a key role in facilitating the integration of patterns into learning models, as they provide developers with the necessary knowledge and tools to use programming languages effectively. Overall, there is a plethora of learning materials available for TypeScript and C#.

They include online courses, textbooks, documentation, video tutorials, and self-study resources (Table 3). Furthermore, development environments and tools are of great value for creating adaptive learning solutions.

Table 3. Examples of learning resources related to TypeScript and C#

Resource type	TypeScript	C#
Official documentation	TypeScript Documentation	C# Documentation
Online courses	Udemy: Learn TypeScript	Udemy: C# Basics for Beginners
Books	"Pro TypeScript" (Fenton, 2018)	"C# in Depth" (Skeet, 2019)
Video lessons	YouTube: TypeScript Tutorials	YouTube: C# Tutorials
IDE	Visual Studio Code	Visual Studio
Testing tools	Jasmine, Mocha	NUnit, xUnit
Package managers	npm (Node Package Manager)	NuGet
Frameworks	Angular, React	ASP.NET (Active Server Pages.Net), Xamarin

Source: created by the authors

Among the available learning materials and tools, the official documentation for TypeScript and C# is particularly important. They provide the most updated information about the programming languages, including new features, practical examples, and best practice guidelines. Books are also valuable resources. They offer a deep dive into topics related to the features and capabilities of the languages. YouTube video tutorials on TypeScript and C# can also be a useful addition, as they provide a visualisation of the learning process and help students to understand the material better.

In terms of tools and platforms, Visual Studio Code for TypeScript and Visual Studio for C# are the best development environments available. Both IDEs offer a user-friendly interface and powerful features that facilitate efficient software development. Frameworks such as Angular for TypeScript and ASP.NET for C# open new horizons for building web applications, which is critical in the context of responsive learning solutions. Furthermore, testing tools such as Jasmine and Mocha for TypeScript and NUnit and xUnit for C# ensure high quality code. They allow detecting errors at early stages of development, which is essential for maintaining the stability and reliability of learning systems. All these resources and tools contribute to the effective implementation of design patterns in adaptive learning models, which makes learning more personalised and effective.

Thus, the study found that the use of design patterns together with the typed programming languages TypeScript and C# considerably increases the effectiveness of adaptive models for personalised learning. The implementation of patterns such as Strategy and Singleton demonstrates their ability to flexibly manage learning content and user data, which ensures individualisation of the learning process. Available learning resources, tools, and powerful development environments, such as Visual Studio Code and Visual Studio, facilitate the effective implementation of these patterns, which makes learning more adaptive, personalised, and focused on the needs of each student.

DISCUSSION

This study found that the use of Singleton, Strategy, Observer, and Factory patterns in combination with TypeScript and C# allows creating adaptive and individualised training models that increase the flexibility and reliability of the system. M. Tanweer & A. Ismail (2024) examined the use of generative AI to create individualised learning experiences that allow adapting the learning materials to the needs of each student. This approach is in line with the findings of the current study, where the use of Singleton and Strategy patterns helped to personalise content for the user. W. Uriawan *et al.* (2024) used TypeScript to develop a web-based platform for learning programming, which ensures the reliability and scalability of the system. This study confirmed the effectiveness of using TypeScript, as in the present study, where this language contributes to the flexibility of the system through its static typing and support for dynamic interfaces. This makes it easy to adapt the learning content to the individual needs of users. At the same time, S. Wang (2023) showed in his study on e-learning software development in C# that content adaptation can improve learning effectiveness, which is consistent with the findings of the current study on the effectiveness of C# for creating stable adaptive systems. Thus, the study complements the findings of S. Wang (2023), confirming the effectiveness of C# in the development of personalised learning platforms focused on supporting individual user needs.

On the other hand, S.S. Khowaja *et al.* (2020) highlighted the value of crowdsourcing to optimise the choice of design patterns, which is consistent with the current study, where the use of such patterns increases the flexibility of the adaptive system. Thus, the current study corroborated the findings of S.S. Khowaja *et al.* (2020) by showing that the integration of Singleton and Strategy patterns also contributes to the personalisation of content for the user, which enhances the effectiveness of the adaptive model. S. Huang *et al.* (2024) developed a recommendation system that considers users' interests in real time, which is consistent with the findings of the current study on increasing individualisation through the Strategy and Observer patterns. Thus, the present study confirmed the findings of S. Huang *et al*. (2024) by demonstrating the effectiveness of using patterns to adapt content to the user's needs in dynamic learning environments.

The findings of A. Blažić *et al*. (2024) demonstrated a learning model that integrates AI to individualise learning, which confirms the value of an adaptive approach in learning systems. This correlates with the current study, which focused on increasing the flexibility and adaptability of the system through design patterns, which ensures a personalised approach to the needs of each user. Furthermore, M. Rahman *et al*. (2023) investigated the automated detection of design patterns that improve the efficiency of object-oriented systems. This is in line with the current approach, where the use of design patterns facilitates the structured adaptation of learning materials. In other words, the findings of both studies confirm the effectiveness of patterns in improving the efficiency and flexibility of adaptive systems.

H. Zhang *et al*. (2024) applied the Adaptive Ensemble C-learning method to improve the performance of systems with real agents, which illustrated the possibility of adapting learning systems for concrete tasks. This is also in line with the current study, where the use of design patterns helped to adapt learning content to the needs of users, increasing the flexibility and personalisation of the learning process. S. Wang *et al*. (2024) discussed the use of simulations in learning using C#, which confirms the choice of C# in the current study to ensure stability and integration with learning content. In this study, C# was used to develop adaptive learning systems, particularly for efficient data management and high performance through the support of Generics and LINQ capabilities. This enables a more comprehensive adaptation of content and integration with a variety of design patterns than in the study by S. Wang *et al*. (2024), which allows for more flexible and scalable learning systems.

Moreover, H. Washizaki *et al*. (2022) showed that design patterns can help integrate machine learning into adaptive learning systems. This is in line with the results of the present study, where such patterns contribute to the adaptability of the model. Therefore, the findings of the current study partially confirm the conclusions of H. Washizaki *et al*. (2022), as they also showed that the use of design patterns greatly improves the adaptability of learning systems. The findings of A.O. Dagunduro *et al*. (2024) emphasised that adaptive learning models promote educational equity, which is also a significant aspect of the current study, which demonstrated the individualisation of learning materials. The difference between the approach in this study is the emphasis on the use of design patterns that enable flexible adaptation of content to meet the particular needs of each student, which increases learning efficiency and promotes a personalised approach to each user.

M.K. Chong (2021) developed a TypeScript-based group project platform, which proved to be effective in ensuring interactivity and flexibility. Analogously, in the current study, TypeScript was chosen for its ability to create robust and responsive web platforms capable of rapid integration and adaptation. Although both studies employed comparable technologies, the current study also considered complementary aspects of content adaptation and architectural flexibility. M.E. Attia & M.A. Arteimi (2021) used fuzzy logic in Moodle to individualise learning, which supports the findings of the current study on the value of adapting learning content to the needs of the user. The authors' fuzzy logic approach enabled the creation of flexible and adaptive systems, which is consistent with the use of design patterns in the current study to improve the flexibility and scalability of adaptive learning platforms.

S. Latif *et al*. (2022) developed an approach to automate the detection of design patterns using machine learning, which is consistent with the current approach to using patterns to structure an adaptive system. Their study with automated pattern detection allows significantly simplifying the system adaptation process to new conditions and user requirements. In the current study, the use of design patterns also contributed to the flexibility and adaptability of the system, but with an added emphasis on personalising learning content, which extends the capabilities of the approaches proposed in other studies. F. Gnadlinger *et al*. (2023) proposed a system for optimising the management of object-oriented applications based on design patterns, which confirms the current findings on a structured approach to the development of personalised learning models. In both cases, patterns increase flexibility, reduce system complexity, and enable a more efficient customisation to meet specific user requirements.

D.M. Arya *et al*. (2024) showed adaptive learning models, which emphasises the importance of integrating AI to individualise learning content. This confirms the results of the present study, where the use of technology to adapt learning materials can increase the effectiveness of learning, considering the needs of students. The common aspects of both studies were the use of C# and TypeScript programming languages. However, in the current study, these languages were applied with a focus on their ability to create adaptive systems, particularly through the use of typed languages to ensure the reliability, scalability, and flexibility of learning platforms. As in the present study, K. Chen *et al*. (2024) addressed design patterns such as Singleton, Factory, Observer, and Strategy, as they facilitate effective model management, deployment strategies, and team collaboration. These patterns are also instrumental in creating adaptive and scalable systems, which is in line with the findings of the current study on their role in increasing the flexibility and personalisation of learning platforms.

Additionally, R. Xu *et al.* (2024) confirmed that adaptive models are key to the development of effective learning systems, which supports the findings of the current study on the effectiveness of using patterns to adapt learning content. Furthermore, the emphasis of the cited study on the use of design patterns to integrate machine learning is consistent with the approach of the current study to use such patterns to increase the flexibility and adaptability of learning systems. A. Er-Rafyg *et al.* (2024) theorised the significance of empowering adaptive platforms to support a personalised approach, which is consistent with the current study where design patterns improved content adaptation. The concept of individualising learning through platform adaptation offered by A. Er-Rafyg *et al.* (2024) confirmed the effectiveness of using design patterns to provide a personalised user experience in the current adaptive system.

Overall, the analysis of related studies supported the significance of using design patterns and typed programming languages to build adaptive learning models. Studies point to the key role of patterns such as Singleton, Factory, Strategy, and Observer in increasing the flexibility, adaptability, and efficiency of systems. The use of programming languages such as C# and TypeScript allows for high reliability, scalability, and integration with learning content, which ensures that learning is customised to the needs of users. Overall, the present study complemented these findings by proposing novel approaches to content adaptation and management of learning systems using proven methods and technologies.

CONCLUSIONS

The study confirmed the effectiveness of using design patterns and typed programming languages to create adaptive learning models. Based on the analysis of design patterns, namely Singleton, Factory, Strategy, and Observer, it was found that their use increases the flexibility, adaptability, and scalability of learning systems. The use of C# and TypeScript programming languages ensures the reliability and stability of adaptive platforms, enabling effective integration of learning content and adaptation strategies. As a result, the developed software prototypes have demonstrated an increase in the effectiveness of personalised learning using these technologies, which allows accommodating individual student needs and dynamically adapting content to various user requirements.

The obtained findings suggest that the tools and methods proposed in this study contribute to a major improvement in the adaptability of learning models. They allow maintaining high learning efficiency by increasing the degree of personalisation and providing flexible customisation for diverse types of users. The use of design patterns, particularly to guide adaptive strategies, enables more sustainable and flexible learning systems that can be easily expanded or modified to meet new requirements. The findings also showed that typed programming languages are effective in integrating different components and ensuring their stable operation.

Limitations of the study include the fact that it considered in detail the use of a limited number of design patterns and programming languages to build adaptive systems. Furthermore, while the findings demonstrated improvements in the adaptability and effectiveness of learning platforms, some aspects of the interaction between various design patterns or programming languages were not explored, which could affect other aspects of system adaptability.

Recommendations for further research include expanding the analysis of design patterns for adaptive learning systems, including the use of the latest AI techniques to automate content adaptation. Furthermore, it is advisable to explore other types of typed programming languages that may have advantages in the context of adaptive systems. It is also vital to investigate the effect of integration with other educational platforms, which allows creating more versatile solutions for diverse learning contexts.

ACKNOWLEDGEMENTS

None.

CONFLICT OF INTEREST

None.

REFERENCES

[1] Arya, D.M., Guo, J.L.C., & Robillard, M.P. (2024). Properties and styles of software technology tutorials. *IEEE Transactions on Software Engineering*, 50(2), 159-172. doi: 10.1109/TSE.2023.3332568.

[2] Attia, M.E., & Arteimi, M.A. (2021). Adaptive e-learning system using fuzzy logic. *Al Academia Journal for Basic and Applied Sciences (AJBAS)*, 3(3).

[3] Ball, T., de Halleux, P., & Moskal, M. (2019). Static typescript: An implementation of a static compiler for the typescript language. In *Proceedings of the 16th ACM SIGPLAN international conference on managed programming languages and runtimes* (pp. 105-116). New York: Association for Computing Machinery. doi: 10.1145/3357390.3361032.

[4] Blažić, A., *et al.* (2024). Development of the adaptive learning concept at CARNET. In *Proceedings of the 15th international conference on e-learning*. Belgrade: Belgrade Metropolitan University.

[5] Chai, L., Yu, W., & Zhou, N. (2024). Personalized federated learning with adaptive information fusion. *The Journal of Supercomputing*. doi: 10.21203/rs.3.rs-4598644/v1.

[6] Chen, K., *et al.* (2024). Deep learning and machine learning: Advancing big data analytics and management with design patterns. *arXiv (Cornell University)*. doi: 10.48550/arXiv.2410.03795.

[7] Chong, M.K. (2021). *E-learning platform for collaborative coding assignments*. (Doctoral dissertation, Universiti Tunku Abdul Rahman, Kampar, Malaysia).
[8] Dagunduro, A.O., Chikwe, C.F., Ajuwon, O.A., & Ediae, A.A. (2024). Adaptive learning models for diverse classrooms: Enhancing educational equity. *International Journal of Applied Research in Social Sciences*, 6(9), 2228-2240. doi: 10.51594/ijarss.v6i9.1588.
[9] Dumitru, C.T. (2024). Future of learning: Adaptive learning systems based on language generative models in higher education. In S. Tripat & J. Rosak-Szyrocka (Eds.), *Impact of artificial intelligence on society* (pp. 33-44). New York: Chapman and Hall. doi: 10.1201/9781032644509-3.
[10] Er-Rafyg, A., Zankadi, H., & Idrissi, A. (2024). AI in adaptive learning: Challenges and opportunities. In A. Idrissi (Ed.), *Modern artificial intelligence and data science* (pp. 329-342). Cham: Springer. doi: 10.1007/978-3-031-65038-3_26.
[11] Fenton, S. (2018). *Pro TypeScript: Application-scale JavaScript development*. Basingstoke: Apress. doi: 10.1007/978-1-4842-3249-1.
[12] Gnadlinger, F., Selmanagic, A., Simbeck, K., & Kriglstein, S. (2023). Adapting is difficult! Introducing a generic adaptive learning framework for learner modeling and task recommendation based on dynamic Bayesian networks. In *Proceedings of the 15th international conference on computer supported education* (pp. 272-280). Prague: SciTePress. doi: 10.5220/0011964700003470.
[13] Gou, Q., & Poliakova, H. (2024). Measurement of personalized learning of students in the digital educational environment of the institution of higher education on a qualimetric basis. *Adaptive Management: Theory and Practice, Series Pedagogics*, 18(35). doi: 10.33296/2707-0255-18(35)-19.
[14] Halkiopoulos, C., & Gkintoni, E. (2024). Leveraging AI in e-learning: Personalized learning and adaptive assessment through cognitive neuropsychology – a systematic analysis. *Electronics*, 13(18), article number 3762. doi: 10.3390/electronics13183762.
[15] Huang, S., Yang, H., Yao, Y., Lin, X., & Tu, Y. (2024). Deep adaptive interest network: Personalized recommendation with context-aware learning. *arXiv (Cornell University)*. doi: 10.48550/arXiv.2409.02425.
[16] Khowaja, S.S., *et al.* (2020). Crowdsourced machine learning based recommender for software design patterns. *International Journal of Computer*, 36(1), 34-52.
[17] Koshova, O., Chernenko, O., Chilikina, T., & Komar, I. (2023). Peculiarities of web applications developing for the distance learning system using the react library. *Systems and Technologies*, 65(1), 20-31. doi: 10.32782/2521-6643-2023.1-65.3.
[18] Latif, S., Qureshi, M.M., & Mehmmod, M. (2022). Detection and recognition of software design patterns based on machine learning techniques: A big step towards software design re-usability. In D.N.A. Jawawi, I.S. Bajwa & R. Kazmi (Eds.), *Engineering software for modern challenges* (pp. 3-15). Cham: Springer. doi: 10.1007/978-3-031-19968-4_1.
[19] Mirzaei, M., & Meshgi, K. (2023). The use of machine learning in developing learner-adaptive tools for second language acquisition. In *CALL for all languages – EUROCALL 2023 short papers* (pp. 272-277). Reykjavik: University of Iceland. doi: 10.4995/EuroCALL2023.2023.16996.
[20] Peng, P., & Fu, W. (2022). A pattern recognition method of personalized adaptive learning in online education. *Mobile Networks and Applications*, 27(3), 1186-1198. doi: 10.1007/s11036-022-01942-6.
[21] Pravorska, N., & Hryha, S. (2024). Methods for implementing microservice architectures: Advantages and disadvantages, implementation and testing in the development of software applications. *Herald of Khmelnytskyi National University. Technical Sciences*, 335(3(1)), 404-408. doi: 10.31891/2307-5732-2024-335-3-55.
[22] Rahman, M., Hossain Chy, S., & Saha, S. (2023). A systematic review on software design patterns in today's perspective. In *Proceedings of the 11th international conference on serious games and applications for health* (pp. 1-8). Athens: IEEE. doi: 10.1109/SeGAH57547.2023.10253758.
[23] Skeet, J. (2019). *C# in depth*. London: Manning.
[24] Tanweer, M., & Ismail, A. (2024). Generative AI in curriculum development: A framework for adaptive, customized, and personalized learning. In Z. Fields (Ed.), *Impacts of generative AI on creativity in higher education* (pp. 197-230). New York: IGI Global Scientific Publishing. doi: 10.4018/979-8-3693-2418-9.ch008.
[25] Troussas, C., Krouska, A., & Sgouropoulou, C. (2020). A novel teaching strategy through adaptive learning activities for computer programming. *IEEE Transactions on Education*, 64(2), 103-109. doi: 10.1109/TE.2020.3012744.
[26] Uriawan, W., Putra, R.D., Siregar, R.I., Gunawan, S.N., Adriansyah, S., & Nurrohman, W. (2024). BrainNest: Implementation of TypeScript and MERN stack to improve scalability of interactive and personalized e-learning. *Preprints*. doi: 10.20944/preprints202407.0051.v1.
[27] Wang, S. (2023). Developing and implementing effective e-learning software for mechanics: A study of FET and C#. In *Proceedings of the 5th international workshop on artificial intelligence and education* (pp. 125-130). Tokyo: IEEE. doi: 10.1109/WAIE60568.2023.00030.

[28] Wang, S., Mao, X., & Zhang, Y. (2024). Development of e-learning software for aluminum alloy bending experiment based on simulation technology. In *Proceedings of the 5th international conference on computer science, engineering, and education* (pp. 39-44). Shanghai: IEEE. doi: 10.1109/CSEE63195.2024.00016.
[29] Washizaki, H., Khomh, F., Guéhéneuc, Y.-G., Takeuchi, H., Natori, N., Doi, T., & Okuda, S. (2022). Software-engineering design patterns for machine learning applications. *Computer*, 55(3), 30-39. doi: 10.1109/MC.2021.3137227.
[30] Xu, R., Zhang, L., & Chollathanrattanapong, J. (2024). A study of the adaptability of adaptive learning systems to individualized educational strategies. *Applied Mathematics and Nonlinear Sciences*, 9(1). doi: 10.2478/amns-2024-2737.
[31] Zhang, H., Lin, Y., Shen, S., Han, S., & Lv, K. (2024). Enhancing off-policy constrained reinforcement learning through adaptive ensemble C estimation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(19), 21770-21778. doi: 10.1609/aaai.v38i19.30177.

Використання патернів проектування та типізованих мов при розробці адаптивної моделі персоналізованого навчання

Павло Федорка
Доктор філософії, доцент
Ужгородський національний університет
88000, пл. Народна, 3, м. Ужгород, Україна
https://orcid.org/0000-0002-9242-5588
Федір Сайберт
Магістр, асистент
Ужгородський національний університет
88000, пл. Народна, 3, м. Ужгород, Україна
https://orcid.org/0009-0004-8081-4174
Роман Бучук
Кандидат фізико-математичних наук, доцент
Ужгородський національний університет
88000, пл. Народна, 3, м. Ужгород, Україна
https://orcid.org/0009-0000-4199-5583

Анотація. Мета роботи полягала у визначенні ефективності застосування шаблонів проєктування та типізованих мов програмування, зокрема TypeScript і C#, у побудові адаптивної моделі персоналізованого навчання у сфері програмної інженерії. Під час дослідження було розглянуто використання шаблонів проєктування при розробці адаптивної моделі персоналізованого навчання, проведено огляд та використання мов TypeScript та C# у створенні такої моделі, а також порівняно дані типізовані мови програмування та ресурси для навчання у програмній інженерії. Основні результати дослідження показали, що серед шаблонів проєктування найефективнішими для побудови адаптивної моделі персоналізованого навчання є Singleton, Factory, Strategy та Observer, оскільки вони підвищують гнучкість і адаптивність системи. Розроблені програмні прототипи продемонстрували, що використання мови TypeScript забезпечує надійність адаптивної системи завдяки статичній типізації та гнучким інтерфейсам, а мова C# з можливостями Generics та Language Integrated Query (LINQ) сприяє ефективному управлінню даними та модульною інтеграцією. У порівняльному аналізі виявлено, що мова C# краще підходить для складніших систем з високими вимогами до управління даними, тоді як TypeScript забезпечує швидку інтеграцію й більшу гнучкість у розробці фронтенду. Також проведений огляд доступних навчальних ресурсів для обох мов виявив більшу різноманітність для TypeScript, що може сприяти швидшому освоєнню для нових користувачів. Висновки свідчать, що застосування шаблонів проєктування та типізованих мов програмування є важливим підходом до створення персоналізованих навчальних моделей, що здатні адаптуватися до індивідуальних потреб користувача та підвищувати ефективність навчання у програмній інженерії

Ключові слова: індивідуалізована освіта; диференційоване навчання; програмна архітектура; об'єктно-орієнтоване моделювання; налаштовувані інтерфейси; оптимізація даних