Universidad de Guadalajara

Centro Universitario de los Valles

Software Configuration Management

Final Project

Title of the Project:

Web Application Development Project for Educational Management

Master: Software Engineering

Third semester

Nerina Peña Olivero 222977407

Content presented

The content presented in this document is distributed as follows:

Introduction

A general idea is given of the elements of Software configuration management and how they will be analyzed in the context of the project in question.

Chapter 1: Project Baselines

This chapter states the objectives of the project, the future modules that are intended to be implemented to solve the problem as well as the most general requirements. One section contains all the functional requirements of the system corresponding to each module. A following section includes the non-functional requirements of the system classified according to quality characteristics and implementation requirements.

Another section is dedicated to requirement analysis and another dedicated to system design elements where some diagrams of the design stage are presented.

Chapter 2: Change Requests

This chapter presents the change analysis of 4 change requests proposed for the project. The analysis includes elements such as: modules or elements that are affected by the change, and implications of these changes from the point of view of time, budget, effort and human resources.

Chapter 3: Policy for decision making

This chapter presents in detail the policy defined for making decisions regarding change requests. The functions of the members of the approval committee are described, the decision-making process is described, guidelines for deciding and criteria that should be taken into account to approve changes.

Chapter 4: Approval of change requests

Based on the analysis presented in Chapter 2, the analysis is carried out for the approval of the proposed change requests, said analysis and the justification for approval or not of the changes are presented. Also presented in this chapter are the changes in the project's

baselines and a guide to perform audits of the baseline configuration and based on this guide, the results of the audit of some elements in the context of this project.

Chapter 5: Status accounting

This chapter presents a Software Configuration Management policy regarding project accounting status which establishes clear guidelines for measuring, controlling and adjusting resources, such as time, money, effort and team skills, after estimating. At the end of the chapter, the current state of project status accounting is analyzed.

Conclusions

The importance of software configuration management is mentioned, mainly highlighting the lessons learned from carrying out the project as well as the conclusions that could be reached after having carried out the analysis within SCM of real change requests for a project.

Table of contents

Introduction

Software Configuration Management (SCM) is a set of practices and processes used to manage and control configuration items throughout the software lifecycle. The main objective is to guarantee the integrity and quality of the software, as well as to facilitate effective collaboration in development teams. Some key aspects of SCM include:

Configuration Identification

This process focuses on identifying and defining the configuration elements in a system. Configuration items can be anything from source code and documents to configuration files. Accurate identification is essential to manage and control software changes.

Configuration items are the basic units that are managed in the software configuration management process.

In the specific case of this project, the selected configuration elements include:

Source Code: The files that contain the source code of a program.

Documentation: Manuals, specifications, and other documents related to the development and use of the software.

Design: Although design is not typically a configuration element, design-related documents, such as diagrams, specifications, and other design artifacts, are considered configuration elements. In this sense we will adopt the design as a configuration element, seeing it from its artifacts including functional and non-functional requirements.

Requirements: Documents that specify the functional and non-functional requirements of the system.

Because this wornk focuses only on the Design and Implementation stage, only these configuration elements are taken but they are not the only ones, each stage of the life cycle can have its configuration elements and each team can select them according to your need for control

Configuration Control

Involves managing and controlling changes to the identified configuration elements. This ensures that any modifications are made in a planned and authorized manner, avoiding unwanted changes and guaranteeing the integrity of the system.

Taking into account the configuration elements selected above (code, documentation and requirements), in the configuration control the time, budget, effort and human resources that will intervene in the project will be analyzed taking these elements into account.

In the context of software configuration management, time, budget, effort, and human resources are generally not considered directly as configuration items, but are critical aspects of the project

that are managed holistically. These are handled in project planning and control, and their management falls within the broader scope of project management. Here is a brief description of how these aspects relate to configuration management:

Time: Time management involves planning and tracking the project schedule. Configuration management can influence timelines by closely monitoring changes to configuration items and ensuring that they do not negatively impact project milestones.

Budget: Budget management involves planning and controlling project costs. Configuration management contributes to cost control by ensuring that configuration changes are managed efficiently and that necessary resources are appropriately allocated.

Effort: Effort management relates to the allocation and control of human resources to carry out project activities. Configuration management contributes to effort control by optimizing change processes and minimizing the impact on the development team.

Human Resources: Human resource management involves planning, procurement and development of the project team. Configuration management can influence the workload and task allocation for the development team.

In summary, although these aspects are not configuration items, they are crucial considerations in software project management, and configuration management helps ensure that these aspects are managed effectively throughout the software lifecycle.

Status Accounting

This element involves keeping track of and documenting the current status of configuration items over time. It helps you understand which specific versions of items are in use and provides a history to track changes and previous versions.

Status accounting in software configuration management focuses primarily on keeping track of and documenting the current state of configuration items over time. This includes information about versions, changes made, and the current status of items.

In the context of status accounting, information related to time, budget, effort, and human resources can be recorded, linking this information to configuration items to have a complete understanding of how these aspects relate to software development.

For this project it will be done as follows:

Time: Taking into account the initial time allocated to the original project, how this time will be affected if the identified configuration elements must change.

Budget: Taking into account the initial budget allocated to the original project, how this budget will be affected by changes or versions of the configuration elements, allowing tracking of the costs associated with each modification.

Effort and Human Resources: It will be documented how much effort was dedicated to each activity associated with changes in configuration elements and how many additional human resources are required to carry it out.

In summary, although status accounting focuses on the aspects of versions and changes in configuration elements, to extend its usefulness we associate it with information on time, budget, effort and human resources to have a comprehensive view of software development.

Auditing

Auditing in the context of software configuration management involves reviewing and evaluating processes and records to ensure compliance with established policies and practices. This may include change review, standards compliance, and the overall effectiveness of the configuration management process.

Audits can be carried out at any time in the life cycle of a software development project. Some of the key moments to perform SCM audits include:

Baseline Configuration Audit: Performed after establishing a software baseline. This involves auditing and validating the initial software configuration, including associated documentation.

Continuous Process Audit: Throughout development, periodic audits can be conducted to evaluate adherence to SCM processes. This would include version management, change control and consistency in documentation.

Pre-Delivery Audit: Before delivery of a release or product, an audit is performed to ensure that the software configuration meets requirements and is ready for delivery.

Post-Implementation Audit: After implementation, an audit can be performed to evaluate the impact of changes, identify lessons learned, and improve SCM processes for future developments.

In the specific case of this project, Baseline Configuration Audit was chosen to present a guide on what to audit and how to do it.

Chapter 1: Project Baselines

These are the baselines of the initial project, before change requests

Project's name: Web Application Development Project for Educational Management

Project objective: Develop a web application that allows the efficient management of courses, qualifications and educational activities for a school institution. The application will be designed to address the needs of teachers, parents and students, providing specific modules for each user group.

Main features

Teacher Module:

- ✓ Register of teachers with personal and contact information.
- ✓ Creation and management of courses.
- ✓ Record of grades and feedback for students.
- ✓ Scheduling and management of educational tasks and activities.
- ✓ Notifications about school events, activities and important announcements.
- ✓ Internal communication with other teachers, parents and administrators
- ✓ Generation of reports on the academic performance of students in their courses.

Parent Module:

- ✓ Registration of parents or guardians with relevant information.
- ✓ Access to your children's grades and academic progress.
- ✓ Notifications about school events, activities and important announcements.
- ✓ Communication with teachers and other parents through the platform.

Student Module:

- ✓ Student registration with personal and contact information.
- ✓ Visualization of class schedules and course details.
- ✓ Access to grades, progress reports, comments and feedback from teachers.
- ✓ Upload of tasks and digital deliveries.
- ✓ Participation in forums and academic discussions.

Administration Module:

- ✓ User Management.
- ✓ Management of courses, schedules and school calendars.

Functional requirements by modules

Teacher Module:

Teacher Registration:

RF1: Teachers must be able to create user accounts by providing their personal information, such as full name, email address, password, academic grade, teaching grade, subjects taught.

RF2: Teachers should be able to edit their personal and contact information at any time.

RF3: Teachers must be able to log in using their registered credentials.

Creation and Management of Courses:

RF4: Teachers must be able to create new courses by providing details such as name, description and educational level, school grade where it is taught (courses must be associated with the teachers responsible for their teaching).

RF5: Teachers must be able to edit and update existing course information.

Record of Ratings and Feedback:

RF7: Teachers should be able to record student grades on different assignments and tests.

RF8: Teachers must be able to provide written feedback along with grades.

RF9: Teachers must be able to generate reports on the academic performance of students in their courses.

Scheduling and Task Management:

RF10: Teachers should be able to create educational assignments and activities with details like title, description, and due date.

RF11: Teachers should be able to mark assignments complete and assign grades.

Notifications about School Events:

RF12: The system must allow the teacher to send notifications about school events, important dates and planned activities related to their courses.

RF13: Teachers should be able to choose their notification preferences and how they want to receive them (email, messages on the platform, etc.).

Internal communication:

RF14: Teachers must be able to communicate with other teachers, parents and administrators through messages on the platform.

Parent module

Registration of Parents or Guardians with Relevant Information:

RF1: Parents or guardians must be able to register on the platform by providing their name, email address, address, contact telephone number and password. In addition, they must be able to complete their registration with additional information such as the number of children in the school and the student code of their child(ren).

RF2: Parents must be able to login using their registered credentials.

Access to Your Children's Grades and Academic Progress:

RF 3: Parents must be able to access their children's grades and see their academic progress on the platform.

RF4: Parents must be able to view the grades of different subjects and activities to assess their children's performance.

RF5: Parents should be able to view a grade history and summary of progress over time.

Notifications about School Events, Activities and Important Announcements:

RF6: Parents should receive push notifications about school events, key dates, and important announcements.

RF7: Parents should be able to choose their notification preferences and the way they want to receive them (email, messages on the platform, etc.).

Communication with Teachers and Other Parents Through the Platform:

RF9: Parents must be able to communicate with their children's teachers through messages on the platform.

RF10: Parents must have the ability to participate in forums and discussions with other parents to share information and experiences.

Student module

Student Registration with Personal and Contact Information:

RF1: Students must be able to register on the platform by providing their name, email address, student code, and password.

RF2: Students must be able to login using their registered credentials.

Viewing Class Schedules and Course Details:

RF3: Students must be able to access their class schedules.

RF4: Students must be able to view the details of the courses corresponding to their school year.

Access to Teacher Grades, Comments and Feedback:

RF5: Students must be able to access their grades obtained in homework, exams and other academic activities.

RF6: Students must be able to see the comments and feedback provided by teachers regarding their academic performance.

RF7: Students must be able to view their grades for different subjects and activities to see their performance.

RF8: Students must be able to view a grade history and a summary of progress over time.

Upload of Tasks and Digital Deliveries:

RF9: Students must have the ability to upload assignments and digital submissions in their courses.

Participation in Forums and Academic Discussions:

RF10: Students must be able to participate in forums and academic discussions related to the courses.

RF11: Students must be able to post questions, comments and answers in the forums, encouraging collaboration and knowledge sharing.

Administration Module

User Management:

RF1: The system must allow the creation of user accounts for teachers, parents and students.

RF2: Users must be able to log in to the platform using their registered credentials.

RF3: Users must be able to recover forgotten passwords through a secure password reset process.

RF4: The system must validate the uniqueness of email addresses to avoid duplicate registrations.

RF5: The system must verify the authenticity of the email address during the registration process.

RF6: Edit users

RF7: Delete user

Management of Courses, Schedules and School Calendars:

RF8: Administrators must have the ability to create courses on the platform.

RF9: Administrators must have the ability to edit courses on the platform.

RF10: Administrators must have the ability to delete courses on the platform.

RF11: Administrators should be able to schedule class times, assign classrooms, and update schedules as needed.

RF12: Administrators must be able to assign classrooms.

RF13: Administrators should be able to update schedules as needed.

Non-functional requirements

Security and Privacy:

- ✓ To access the system, robust authentication and authorization mechanisms must be in place to guarantee adequate access to information.
- ✓ The system must allow the protection of personal data according to regulations and privacy standards.

Design:

- ✓ Responsive user interface for use on mobile, tablet and desktop.

Notifications and Communication:

- ✓ The system must allow notifications by email or messages within the platform to keep users informed.

Performance:

- ✓ Loading time should be adequate to avoid user frustration.

Availability:

- ✓ The system must be available most of the time so that users can access it when they need it.

Scalability:

- ✓ It must be able to support an increase in the number of users without degrading performance.
- ✓ It should be possible to increase system resources as needed.

Usability:

- ✓ The user interface should be intuitive and easy to navigate for all types of users (teachers, parents, students).
- ✓ The application must comply with web accessibility standards to ensure that all users can use it.

Maintainability:

- ✓ The code should be well structured in modules to facilitate future updates and changes.
- ✓ There should be detailed documentation on the code and architecture to facilitate maintenance by future teams.

Browsers and devices:

- ✓ The app must work consistently across different web browsers and devices.

Requirements analysis

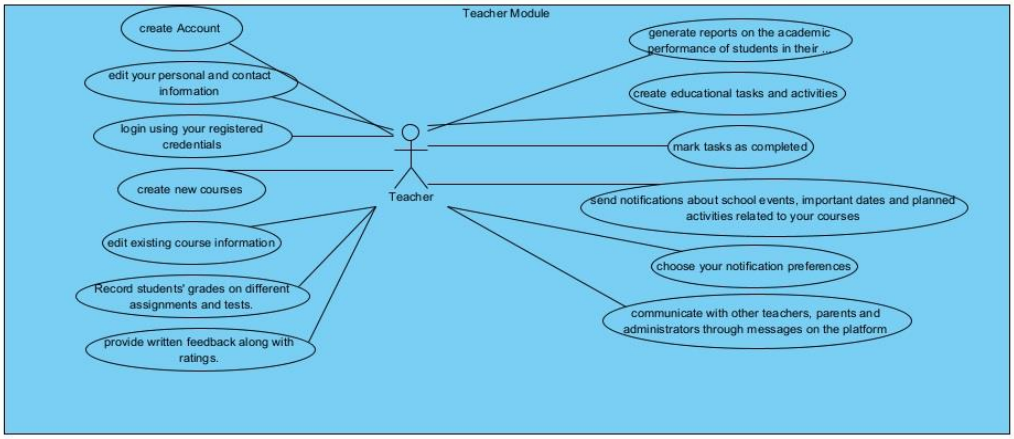
System Use Case Diagrams

A system use case diagram represents the interactions between a system and its external actors, showing how use cases are used to achieve specific objectives. It helps to understand the main functions of the system from the user's perspective and how they interact with it. Each ellipse in the diagram represents a use case, describing a function or feature of the system, while the lines connect actors to the use cases, illustrating interactions.

Below are the use case diagrams for the system in question, one diagram is presented for each module of the system.

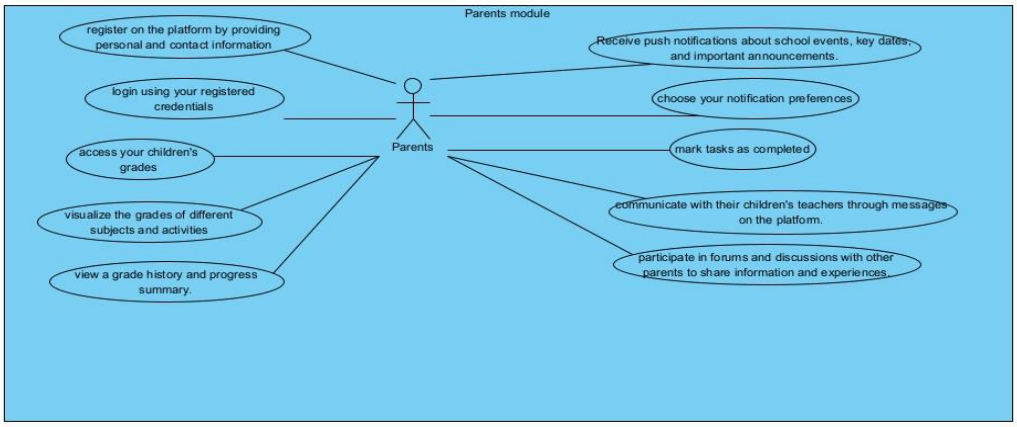
System Use Case Diagram for the Teacher Module:

Main Actor: Teacher



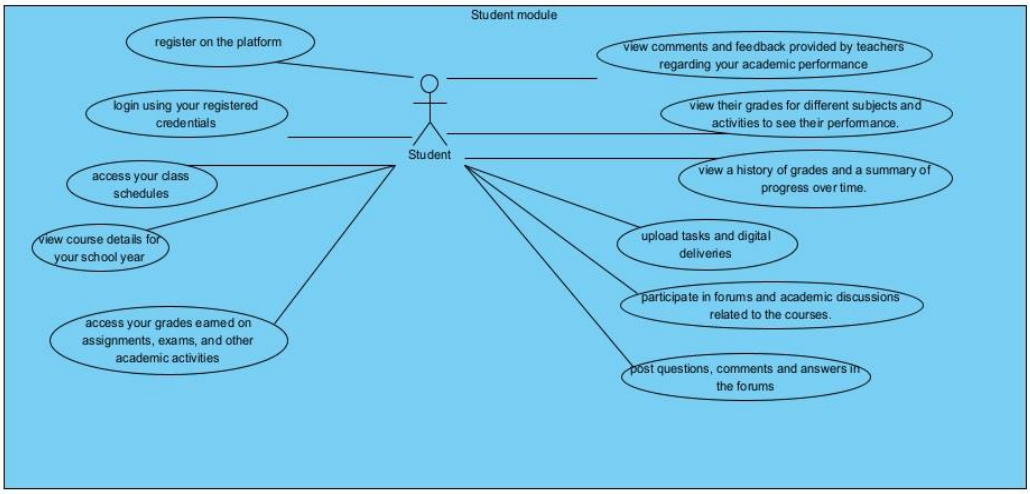
System Use Case Diagram for the Parents Module:

Main Actor: Parents



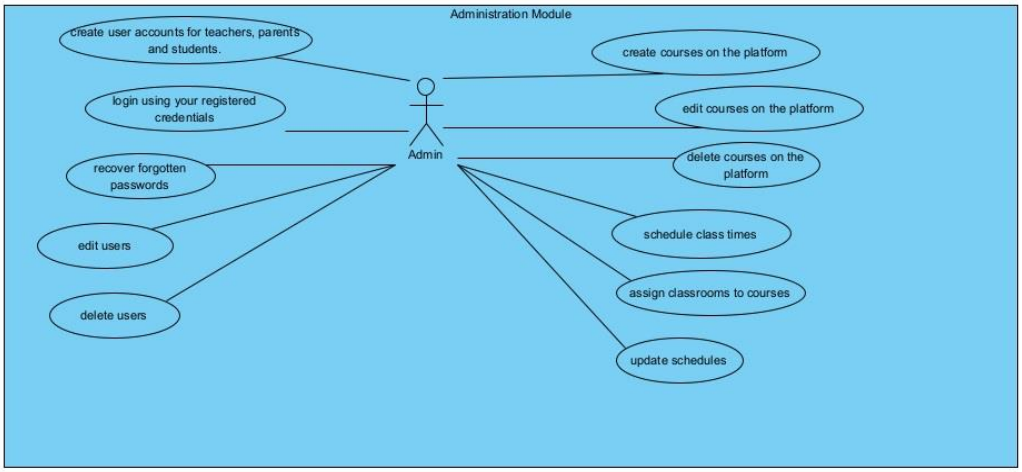
System Use Case Diagram for the Student Module:

Main Actor: Student



System Use Case Diagram for the Administration Module:

Main Actor: Admin



Description of critical system functionalities

The requirements analysis includes a detailed description of each functional requirement in terms of use cases for the system. Due to the size of the system, only some cases of critical uses for the system will be selected, so that there is a representation of what a complete requirements analysis would be like.

ID	01
Name	Create new courses
Actors	Teachers
Description	Teachers should be able to create new courses by providing details such as name, description, and education level, grade level (courses should be associated with the teachers responsible for teaching them).

Use Case Flow	1. The system displays an interface so that the teacher can register the data of the new course, the fields to complete are: course name, description, educational level (drop-down menu), school grade (drop-down menu). 2. The teacher fills in the information and presses the button Save course. 3. The system verifies that all the data is complete, that the data is valid and that the teacher does not have another course with the same values. If everything is correct, the course is saved in the DB and displays a success message. If any data is filled with invalid values, the system will not save the course and will indicate the value that must be corrected 4. Finish the use case.
Specific requirements	RF4 Teacher Module
Preconditions	The teacher must be authenticated in the system.
Postconditions	The new course must be saved in the DB.

ID	02
Name	Access to qualifications
Actors	Parents
Description	Parents should be able to access their children's grades and view their academic progress on the platform.
Use Case Flow	1. The system displays an interface so that the parent can view grades previously recorded by a teacher. 2. The parent must select a course and press the Show Grades button. 3. The system verifies which course was selected and the grades recorded in that course for your child are printed on the screen. 4. Finish the use case.
Specific requirements	RF3 Parent Module
Preconditions	The parent must be authenticated in the system.
Postconditions	The system should display the parent options menu

ID	03
Name	Creating user accounts
Actors	Administrators
Description	The system must allow the creation of user accounts for teachers, parents and students.
Use case flow	1. The system shows an interface so that the admin can register the data of the new account, the fields to complete are: first select the type of user (teacher, parent, student), depending on the type of user, this will be the data to to complete. (See RF of user registration of each module).

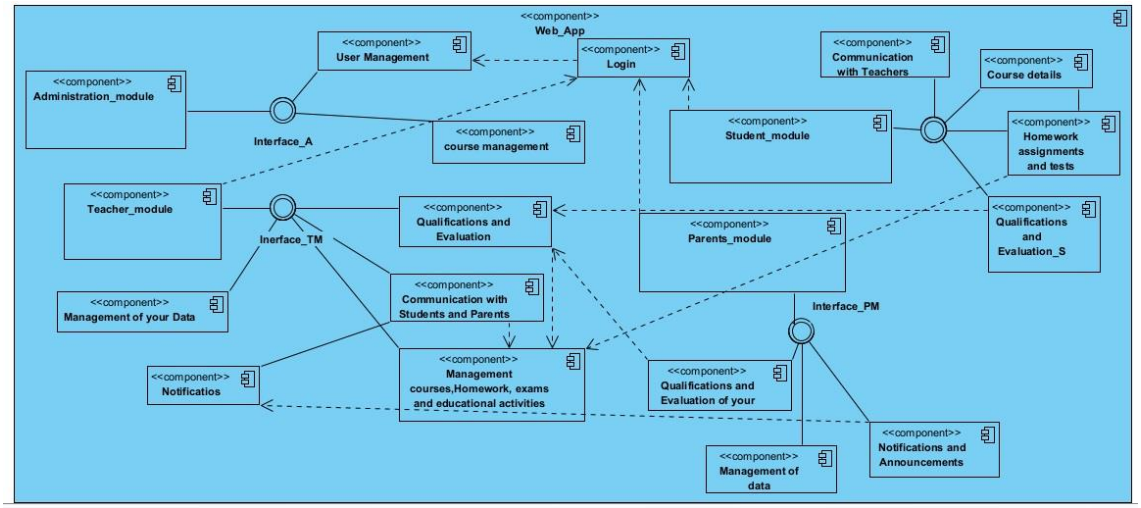
	2. The administrator must complete the information and press the Register User button. 3. The system verifies that all the data is complete, that the data is valid and that there is no other user with the same email address. If everything is correct, the user is saved in the DB and shows a success message. If any data is filled with invalid values or the mail already exists in the database, the system will not save the user and will indicate the value that must be corrected. 4. Finish the use case.
Specific requirements	RF1 Administration Module
Preconditions	The administrator must be authenticated in the system.
Postconditions	The system must save the new account created in the DB

ID	04
Name	Upload tasks and digital deliveries
Actors	Students
Description	Students must have the ability to upload digital assignments and submissions in their courses.
Use Case Flow	1. The system displays an interface for the student to upload their assignment, it will allow them to select the file from a location on their device 2. The student must upload the corresponding file and select the Send button. 3. The system checks that the selected file is in the defined format (pdf , doc , jpg , zip , rar) and that the size does not exceed the allowed amount. If both conditions are met, the file is sent, otherwise the system will throw an error message. 4. Finish the use case.
Specific requirements	RF9 Student module
Preconditions	The student must be authenticated in the system.
Postconditions	The system must save the assignment uploaded by the student in the DB

System design

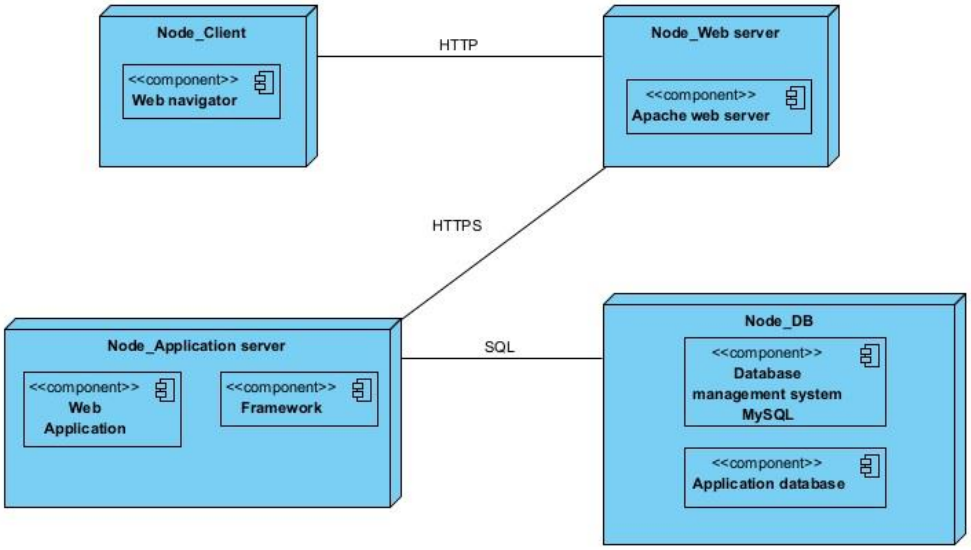
Component diagram

This component diagram describes the elements that make up the web application in terms of components

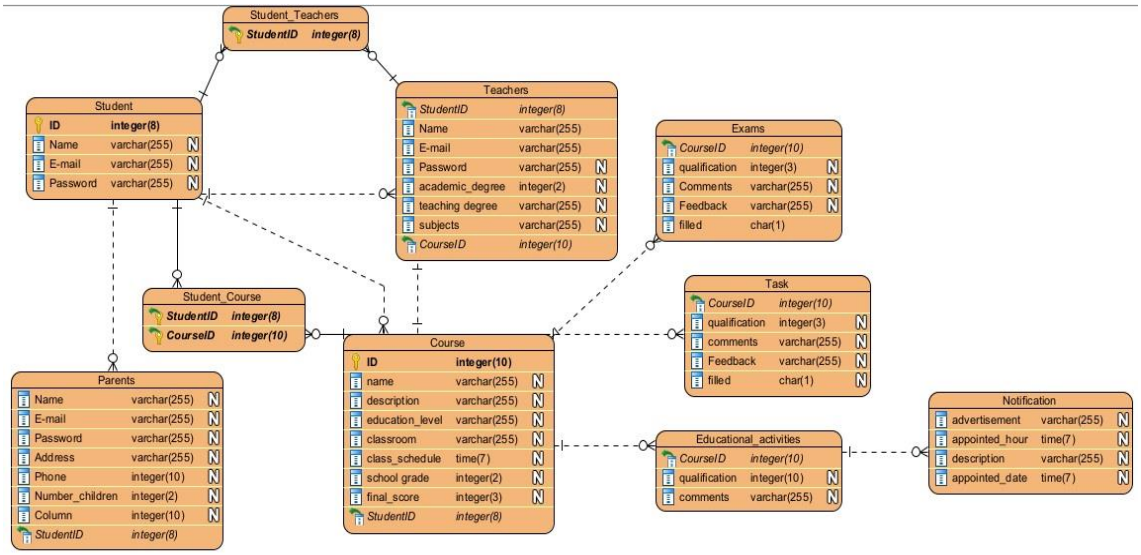


Deployment diagram with components

Below is a componentized deployment diagram that represents how components of a system or application are distributed and communicate in a specific deployment environment. This diagram is representative for a version 1 of the system, in later versions other nodes and components can be included.



Entity relationship diagram



This ER diagram represents the relationship between the database tables of the proposed system.

In the case of tables that are represented by more than one relationship, this representation is given because a table can be related to another table in different ways, including "One-to-Many" and "Many-to-Many" relationships.

For example: "Students" and "Courses". Students can attend multiple courses, which would create a "Many to Many" relationship between students and courses. However, you might also want to keep track of each student's attendance in each course, which would create a "One to Many" relationship between students and courses (a student can have many attendance records, one for each course).

In this case, we have a "Students" table, a "Courses" table, and a third "Students_Courses" table that would relate students to courses and could contain additional information such as date of attendance and other relevant data.

The same happens with the relationship between the Student and Teacher tables that a new table arises from the relationship, which could be used for subsequent system functionalities. Up to this point in the design, only the functionalities listed above are being contemplated, the database design will be prepared for these new functionalities, if they are required in the future.

Chapter 2: Change Requests

Important points

The effort is to be measured in terms of the identified configuration elements

CODE

Adding additional functionality will require writing additional code to implement the forms, database modifications, and report generation. Coding effort will increase based on the complexity of these new features and the need to integrate them with existing code.

DESIGN

The design will also require additional effort, as you will have to plan how the new forms will be integrated and how reports will be presented. This involves fine-tuning the user interface, designing the layout of elements on forms, and creating a structure for reports.

DOCUMENTATION

Updating documentation is key to ensuring understanding and proper maintenance of new functionality and reflecting changes. Changes made to the code, modifications to the database, analysis and design diagrams, and how reports are generated should be documented.

The effort in coding, design and documentation will increase in proportion to the complexity of the three new functionalities. Effort in software development and engineering is measured in several ways, I took the following into account:

- **Work Hours (Person-Day or Person-Month):** This is a basic measure in which the effort required is estimated in terms of man-hours or man-days that will be needed to complete a task.
- **User Story or Use Case Points:** This is an agile metric where each user story or use case is scored based on its complexity and perceived effort. The sum of the points in an iteration gives an estimate of the effort required.

General scheme for assigning points to complexity. Based on the experience of a team

Minimal Complexity (1 point): This category is used for tasks that are very simple and do not require a lot of time, effort, or resources. They can be completed quickly and easily.

Little Complexity (2 points): Low complexity tasks are a little more challenging than minimal complexity tasks, but are still relatively simple. They may require a little more time and attention, but they do not present a significant challenge.

Medium Complexity (3 points): Here we are talking about tasks that have a moderate degree of complexity. They require a significant amount of time and effort to complete, and may involve multiple steps or considerations.

Maximum Complexity (4 points): This level of complexity is assigned to tasks that are very complicated and can be challenging to manage. They may require careful planning and execution, involve multiple stakeholders, or present significant technical challenges.

To calculate the effort in percentage terms, the following formula will be used:

- **Effort (%)= (estimated hours + complexity) / 10**

Where Estimated hours is the sum of the hours needed to complete each task, taking into account the previous effort table.

Complexity in the equation is the sum of the complexities of each task.

The value 10 by which the sum is divided is an adjustment value.

The effort tables of each of the change proposals, in addition to providing elements for the final effort calculation, also provide a clear guide of how the identified configuration elements change: code, design (in addition to requirements) and documentation, serving as a basis. for implementation as it specifies the specific tasks that must be performed in the 3 configuration elements identified for each change request.

The analysis is expanded with other important elements for the efficient management of a software project that will also determine the success or failure of a project such as effort, time, budget and human resources.

Change analysis CR1

CR1.

Due to Mexican Government regulations, a monthly report the following data is requested

- Misconduct cases of students
- Students with alcohol and drugs abuse
- Indices of courses failed by students
- Student dropouts

What does this change imply?

1. **Teacher Module**

Teachers, being those who are in most direct contact with students, must be in charge of collecting and recording the information requested from the system

- Visual interface will have to be modified. Create a form to register new data

2. **Design of the database**

The recorded data must be saved, so it is necessary to create a new table in the database

- ❑ Add a new table. Establish relationships with existing tables

3. Administration Module

Once the information has been collected, it is necessary to generate and distribute the report according to the policies established in the educational institution

- ❑ Visual interface will have to be modified. The report must be generated. View the report, print or send by e-mail

EFFORT					
CODE	**Work Hour**	**DESIGN**	**Work Hour**	**DOCUMENTATION**	**Work Hour**
Modification of the application logic to collect and process data related to cases of misconduct, alcohol and drug abuse, course failure rates, and student dropout. **Medium complexity:3**	5h	Design of the user interface for the entry and display of data related to misconduct, substance abuse, course failure rates, and student dropout. **Little complexity:2**	4h	Updated system documentation to reflect changes made. **Little complexity:2**	2h
Development of functions for the automatic generation of the monthly report with the requested data. **Medium complexity:3**	8h	Update the data model to store the new required information. **Little complexity:3**	4h	Creating user manuals to explain how to enter data relevant to the report. **Minimal complexity:1**	1h
Integration of data validation methods to ensure the accuracy of the	6h	Design of workflows for the efficient generation and presentation of the	2h	Detailed documentation of the new functions and processes	2h

information collected **Little complexity:2**		monthly report. **Minimal complexity:1**		implemented to generate the report. **Minimal complexity:1**	

EFFORT=5.2%

TIME

The total estimated time for the original project is 6 months (1440 hours)

Based on the previous effort calculation, the time to make the changes in terms of coding, design and documentation is:

- **34 hours** equivalent to **4 days and 2 hours** represents **2.36%**

BUDGET

The initial budget for the project is $113 528 MXN

Taking into account the cost of one month of scholarship, the additional budget required to make the changes was calculated.

$ 14191 MXN (for months) equivalent to $473 MXN (for day)

For the change, you additionally need:

$2010 MXN equivalent to **4.25 Additional days** represents **1.77%**

HUMAN RESOURCES

Initially, the project was intended for a single programmer to work on, and if it were possible to hire additional human resources, then an additional person to occupy the role of programmer would be sufficient for this change.

Change analysis CR2

CR2.

The client **wants to develop** a cloud version of the system for commercial purposes.

What does this change imply?

Data Security: Ensure the protection of student and teacher information during transmission.

Scalability: Ensuring that the cloud application can grow to handle larger numbers of users and workloads seamlessly.

Data Migration: Successful migration of existing data from on-premises to the cloud is critical to ensuring a smooth transition.

Costs: Manage costs associated with cloud infrastructure and services to ensure long-term business success.

Business Model: Define a solid business model for the cloud version, including how it will be monetized and users will be billed.

EFFORT					
CODE	**Work Hour**	**DESIGN**	**Work Hour**	**DOCUMENTATION**	**Work Hour**
Data Migration: Develop scripts and processes to migrate existing data from the on-premises application to the cloud infrastructure. **Maximum complexity: 4**	24h	Define the architecture of the cloud application, including the selection of key services and components, such as servers, databases, storage services, and load balancing. The architecture design should prioritize scalability, high availability, and security.	12h	Create a document that describes in detail the functional and non-functional requirements of the new cloud version. Includes information on scalability, security, accessibility, and business features **Minimal complexity:1**	4h

Scalable Architecture Design: Create an architecture that is scalable. This involves a load balancing strategy to handle a larger number of users. **Maximum complexity:4**	16h	**Maximum complexity:4**		Prepare a plan that describes the development, implementation and deployment strategy of the cloud version. **Minimal complexity:1**	4h
Cloud Services Integration: Deploy and configure cloud services, such as cloud databases, cloud storage, authentication and authorization services, and monitoring and logging services. **Maximum complexity:4**	16h			Document the data security, privacy, and regulatory compliance policies that will be applied in the cloud version. **Minimal complexity:1**	4h

Effort 10.2 %

TIME

The total estimated time for the original project is 6 months (1440 hours)

Based on the previous effort calculation, the time to make the changes in terms of coding, design and documentation is:

80 hours **equivalent to** 5.56%

BUDGET

The initial budget for the project is 113 528 MXN (473 MX for day)

The additional budget for this change is:

4730 $ MX **equivalent to** 4.17%

HUMAN RESOURCES

Additionally needed:

1 Cloud Architect: to design the cloud infrastructure, select appropriate services, and ensure the scalability, security, and availability of the cloud application.

1 DevOps Developer: to be in charge of the migration of the application to the cloud, the implementation of cloud services and the automation of deployment and operation processes.

1 Cloud Security Specialist: to ensure data security and compliance and to configure access and audit controls.

Total: 3 people

Change analysis CR3

CR3.

For accessibility purposes, the system should have help legends in written and spoken forms, as requested by a new inclusion law taken into effect next month.

What does this change imply?

User Interface (UI): Significant changes to the user interface. It should be clear and designed so that the legends are easily accessible and understandable.

EFFORT					
CODE	**Work Hour**	**DESIGN**	**Work Hour**	**DOCUMENTATION**	**Work Hour**

Add interface elements to display and activate help legends. **Medium complexity:3**	24h	Create and adapt help legend content to make it clear and useful to users. **Medium complexity:3**	24h	Requires providing detailed documentation to end users on how to use the help legends and how to access the spoken versión **Minimal complexity:1**	4h
Implement code that meets accessibility guidelines and perform tests to verify its operation. **Medium complexity:3**	24h	Ensure that the design is compatible with Screen Readers, that it can be read correctly through screen readers, as it helps users with visual disabilities. **Little complexity:2**	4h		
Develop and structure the content of help legends in the code in a clear and accessible way. **Little complexity:2**	4h				

Effort 9.8%

TIME

The total estimated time for the original project is 6 months (1440 hours)

Based on the previous effort calculation, the time to make the changes in terms of coding, design and documentation is:

84 hours **equivalent to** 5.83%

BUDGET

The initial budget for the project is 113 528 MXN (473 MX for day)

The additional budget for this change is:

4966 $ MX **equivalent to** 4.38%

HUMAN RESOURCES

Additionally needed:

1 Front-End Developer: to implement the help legends in the design and functionality of the application.

1 User Experience (UX) Designer – to work on creating the help legends to ensure they are intuitive and useful to users.

1 Sound engineer: to work on recording, editing spoken captions and ensuring sound quality

Total: 3 people

Change analysis CR4

CR4.

The screens in the system **should change** to meet the company image criteria by

adding the logos of the company on the system's screens.

What does this change imply?

Changes in the design and layout of the screens to incorporate the logos and guarantee an attractive and coherent interface.

EFFORT					
CODE	**Work Hour**	**DESIGN**	**Work Hour**	**DOCUMENTATION**	**Work Hour**

Modify style sheets (CSS) to adjust the appearance with the new design. It involves changing colors, fonts, margins, element sizes, etc. **Medium complexity:3**	8h	Redesign the user interface to reflect the company's new look and brand identity. This involves creating sketches, mockups and prototypes of the new screens and visual elements. **Maximum complexity:4**	40h	Design Documentation (Style Guide): Create a document that details design guidelines, including color palette, typography, button styles, icons and other visual elements. **Minimal complexity:1**	4h
Modification of Templates (HTML) to reflect the new design. It includes rearranging elements on the page, adding containers for logos, menus and other design elements. **Little complexity:2**	8h	Choose a color palette and fonts that are consistent with the company image and apply them throughout the design. Make sure colors are accessible and fonts are legible. **Minimal complexity:1**	4h	User Manual Update: update this documentation to reflect changes in design and layout. Users need up-to-date information on how to navigate and use the app in its new look. **Minimal complexity:1**	4h

Effort 8%

TIME

The total estimated time for the original project is 6 months (1440 hours)

Based on the previous effort calculation, the time to make the changes in terms of coding, design and documentation is:

68 hours **equivalent to** 4.72%

BUDGET

The initial budget for the project is 113 528 MXN (473 MX for day)

The additional budget for this change is:

4020 $ MX **equivalent to** 3.54%

HUMAN RESOURCES

Additionally needed:

1 graphic designer

1 web designer

Total: 2 people

Chapter 3: Policy for decision making

Committee Board

Description of general functions within the project

Key Stakeholders: They can be key stakeholders for the project, such as main client, investor or end users, they can also make important decisions by providing feedback and validation of the implemented functionalities.

Quality Engineer: is responsible for ensuring that the software meets quality standards and regulations. Perform tests, identify issues, and collaborate with the development team to ensure software is bug-free for a quality product.

Project Manager: Responsible for defining the product vision, prioritizing features, managing the task backlog, and representing customer needs. Make decisions about which features will be implemented and in what order.

Manager: Responsible for the planning, monitoring and general management of the project. Ensures the team meets established deadlines and budgets, as well as ensuring human resources and logistics.

Software Architect: The software architect is responsible for designing the structure and architecture of the system. He makes important technical decisions, such as the choice of technologies and design patterns, to ensure the technical quality and scalability of the software.

Lead Developer: The team leader is an experienced developer who leads a subteam within the development group. He is responsible for assigning tasks, monitoring progress, and ensuring that coding best practices are followed.

Decision Making Policy for Project Change Requests

Objective:

This policy is intended to establish a clear and effective process for evaluating and making decisions regarding requests for changes to the software development project. It seeks to ensure that changes are managed efficiently and that an appropriate balance is maintained between flexibility and stability of the project.

Participants in decision-making and tasks:

Project Manager: present the analysis of change requests to demonstrate whether or not they are aligned with business objectives.

Manager: present the strategic perspective regarding the logistics required for the change.

Lead Developer: present the evaluation of the technical feasibility of the proposed changes.

Quality Engineer: present the evaluation and quality impact of the proposed changes.

Software Architect: provide your experience in system architecture and design to present the compatibility evaluation of the proposed changes.

Key Stakeholders: present their opinions and points of view regarding the change

Decision-making process

1. Change Request: Any team member or key stakeholder can submit a change request using a specific change request form.
2. Registration and Initial Evaluation: The Project Manager records the change request and distributes it to the relevant team members. The team initially reviews the request to understand its scope and potential impact.
3. Technical Evaluation: The Lead Developer and Software Architect evaluate the technical feasibility of the change and determine if it aligns with the existing architecture and design.
4. Quality Assessment: The Quality Engineer evaluates the quality impact of the proposed change and determines if additional testing is required.
5. Strategic Evaluation: The Manager and Key Stakeholders evaluate the strategic impact of the change in terms of business objectives, budget and deadline.
6. Review Meeting: The team meets to discuss the findings of the technical, quality and strategic evaluations. The risks of change are evaluated and their urgency and criticality determined.
7. Decision: The team makes a decision on the change request. Possible decisions include:

Approval: The change is approved, and its implementation is planned.

Rejection: The change request is rejected.

Postponement: The decision is postponed for future review or for consideration in planning for the next cycle.

8. Communication: All interested parties are informed about the decision made and the next steps.

Measurement of Exchange Risks:

The risks associated with change requests must be measured in terms of urgency and criticality:

Urgent: Changes that must be implemented immediately due to critical issues in production or regulatory requirements.

Critical: Breaking changes that significantly affect the functionality or stability of the system.

Medium: Changes that have a moderate impact on the system and can be planned in the current development cycle.

Low: Minor changes or improvements that have minimal impact on the system and can be planned in future releases.

Guidelines for Deciding **(Point 7)**

Voting to approve, reject, or postpone a change must be done considering multiple factors.

- Complete Change Request Review: Before voting, all team members should carefully review the change request and understand its scope and implications. This includes technical, quality and strategic evaluation.
- Defining Approval Criteria: Establishing clear criteria for approving changes is essential. Criteria should be specific and measurable, and should reflect project objectives and business requirements. This helps ensure that the decision is based on objective data.
- Risk Assessment: Prior to voting, a full risk assessment must be conducted to determine the urgency and criticality of the change. **Urgent or critical changes may require faster and more rigorous approval.**
- Identification of Key Stakeholders: Identify key stakeholders who may be affected by the change and consider their opinions. Stakeholders should be given the opportunity to express their views before the vote.
- Deliberate Voting: During voting, each team member must have the opportunity to express their opinion in a deliberate and reasoned manner. This may include discussions to clarify doubts or concerns before casting a vote.
- Informed Majority: The decision must be based on an informed majority. This means that team members must vote based on available information, established criteria, and risk analysis.
- Postpone if in Doubt: If there is a significant degree of uncertainty or lack of information, consider postponing the vote to obtain more data or conduct additional research.
- Record of Decisions: Carefully document the decisions made and the reasons behind each vote. This provides transparency and a basis for future references.

Clear criteria for approval

Impact on Customer Requirements: Does the change satisfy a valid customer need or request? Is it aligned with the agreed requirements in the contract or project specifications?

Impact on Quality: Does the change improve the overall quality of the software? Does it increase system stability, security or performance?

Technical Feasibility: Is it feasible to implement the change from a technical point of view? Are additional resources or specialized skills required to make the change effectively?

Cost and Resources: Is the change cost-effective and within the project budget? Does it require a significant investment of time or resources?

Impact on Schedule: Does the change affect the project schedule? Can it be done without delaying the delivery of the product?

Compatibility with Existing Architecture: Is the change consistent with the existing architecture and design of the software? Does it introduce conflicts or integration problems?

Associated Risks: What are the potential risks associated with the change? Can it have a negative impact on other components or functionalities of the system?

Added Value: Does the change add significant value to the project or the client? Does it improve the user experience or satisfy a critical need?

Consistency with Business Strategy: Is the change aligned with the strategic objectives of the organization or the client? Does it contribute to achieving the desired results?

Alignment with Project Objectives: Is the change consistent with the project objectives and vision? Does it contribute to meeting planned milestones and deliverables?

Key Stakeholder Support: Do key stakeholders, including the customer, support the change? Have they expressed their approval or disapproval?

History of Similar Changes: Are there similar changes that have been approved or rejected in the past? What lessons can be learned from those experiences?

In short, a vote to approve, reject, or postpone a change to a software development project should be a rigorous, data-driven process. Consideration of technical, quality and strategic information, together with risk assessment and participation of key stakeholders, contributes to making informed decisions that benefit the project as a whole.

This decision-making policy helps ensure that change requests are comprehensively evaluated, considering both technical and strategic and quality aspects, and that informed decisions are made for the benefit of the project and stakeholders.

Chapter 4: Approval of change requests

Analysis for approval

Desirable criteria for considering approval of a change proposal

Budget: 3%

Time: 3%

H.R: 1

Effort: 5%

This table presents a comparison of the results obtained in the analysis of change requests regarding time, effort, budget and human resources with the desirable criteria for approval.

The values in red correspond to measurements above the desirable value and those in blue are in an adequate range.

CR	Budget	Time	H.R.	Effort	Risk Exp.	L. Mandatory	First Approval	Ranking	Final Decision
CR1	**1.77%**	**2.36%**	**1**	**5.2%**	Low	Yes	Approved	1	**Approved**
CR2	4.17%	5.56%	3	10.2 %	High	No	Approved	3	**Rejected**
CR3	4.38%	5.83%	3	9.8%	Medium	Yes	Approved	1	**Approved**
CR4	3.54%	4.72%	2	8%	Medium	No	Approved	2	**Rejected***

Justification of the decision

CR1 and CR3 they are **approved** because they are changes that respond to regulations mandated by law.

CR2 is **rejected** because it exceeds all desirable approval criteria, but it would be useful to perform an economic analysis to verify that the gains from commercializing the application exceed the additional resources needed to carry out the change. If feasible, implementing the change would be considered in a future version of the system.

CR4 CR4 is **rejected for the moment** because it exceeds the desirable approval requirements, but a new analysis with the client should be considered to determine whether or not the company's visual identity requires that the company logos be on all system screens, because in the change request it says "must" but it is not exactly specified that it is a company policy. If it is mandatory in the company's policies, the change must be approved and implemented.

Changes in the baselines taking into account requests for approved changes

CR1

This section will only present the changes that are added to the project's baselines, based on the approved change requests. Only what changes and how it is changed is presented. The complete update is found in a new version of the baselines, available at: https://github.com/Neri9209/Software_Configuration_Management/blob/a5519a1409fa658c25d46c288d0856b8d6519030/V1%20Complete%20Baseline_Web%20Application%20Development%20Project%20for%20Educational%20Management.docx

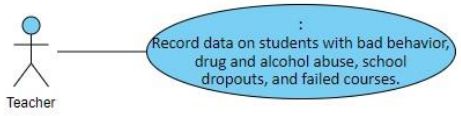
Note: In the project baselines, what is generated from the approval of CR1 appears in green

1. **Teacher Module**

A functional requirement is added to the teacher module:

RF: Record data on students with bad behavior, drug and alcohol abuse, school dropouts, and failed courses.

This requirement implies that from a documentation point of view, a new use case is added, where the Teacher actor interacts:



The description of this use case is added to the baselines of this project.

2. **Design of the database**

Collecting information on students with bad behavior, drug or alcohol abuse, and school dropouts involves creating a new table in the database and establishing a relationship between this new table and one of the existing ones.

New table name: Students_with_difficulties	
Attributes	**Data type**
School dropout	boolean
Drugs abuse	boolean
Alcohol abuse	boolean
School misbehavior	boolean
Failed subjects	boolean

This new table is related to the Courses table, with a One to Many relationship, which means that: A course can have many students with problems. The Courses table in turn is related to the Students table, which indirectly means the Students with problems table is related to the Students table.

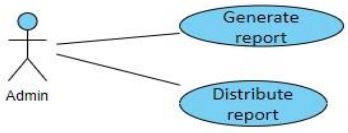
3. **Administration Module**

2 functional requirements are added to the administration module:

RF: Generate report

RF: Distribute report

This requirement implies that from a documentation point of view, 2 new use cases are added, where the Admin actor interacts:



The description of these use cases is added to the baselines of this project.

4. **Other elements that change:**

The design of the interface in the Teacher Module, since a form must be created to collect the data.

The design of the interface in the Administration Module, since new buttons must be created in the menu to Generate reports and to distribute them.

CR3

The approval of the CR3 change request does not imply direct changes in the functional requirements of the system, however when important changes occur in the design of the interface of an application, non-functional requirements are usually impacted, such as performance, usability and accessibility.

It is crucial to consider how these changes will impact system response speed, user experience, and the ability to access and use the interface effectively for different users, including those with disabilities. Additionally, security and maintainability can also be affected, as major changes could introduce vulnerabilities or make the system more difficult to maintain.

Other non-functional requirements were added to the project baselines after approval of CR3, which are available at:
https://github.com/Neri9209/Software_Configuration_Management/blob/9188d507f0889a45a12cb095cb08de4f5650d46d/V2%20Complete%20Baseline_Web%20Application%20Development%20Project%20for%20Educational%20Management.docx

Note: In the project baselines, what is generated from the approval of CR3 appears in purple.

Baseline Configuration Audit

In a baseline configuration audit, several aspects can be audited to ensure the integrity and consistency of the initial software configuration. This section presents, in general terms, some key elements that could be considered to carry out said audit.

The following table presents what to audit and how to do it, as a guide to perform a baseline configuration audit, which, as mentioned in the introduction, this audit was the one selected for the specific case of this project.

The table also presents in the fields marked with gray the results of evaluating these aspects in the context of this project.

Baseline Configuration Audit		
What to check	**How to do it**	**Results of auditing this project**
Configuration Documentation	➢ The documentation associated with the baseline should be reviewed, such as technical specifications, manuals and configuration records. ➢ Ensure documentation accurately reflects current configuration.	Finding: There are no inconsistencies in the documentation; no outdated sections were identified. Recommendation: Keep documentation updated to accurately reflect current configuration.
Version control	➢ Verify that a controlled and labeled version of the baseline has been established. ➢ Ensure that the baseline version corresponds to the associated documentation.	Finding: Versions are tagged (V1 and V2 after initial version) Recommendation: Continue implementing version tags for more effective management. In the initial version of the baselines, before the changes, it should be clearly noted that it is the initial one.
Source Code Integrity	➢ Audit the integrity of the source code that is part of the baseline.	

	➢ Make sure there are no missing files, and that the directory structure is consistent.	
Standards Compliance	➢ Evaluate whether the configuration meets established development standards and best practices (if established).	
Change Log	➢ Review change history to ensure all modifications are documented and justified. ➢ Ensure that changes have been reviewed and approved according to established procedures.	Finding: All changes present documentation and approval. Recommendation: Keep the change documentation and approval process reinforced.
Baseline tests	➢ Verify that tests have been carried out to validate the functionality and stability of the baseline. ➢ Ensure test results are documented and available.	
Security and Access Control	➢ Audit security and access control configuration to ensure adequate protection of the baseline.	Finding: The repository is public and people with the link to the repository have access. There are no collaborators with writing permission, so they will not be able to modify the baselines. Recommendation: Improve security measures; limit access.

The results in the last column of the table briefly summarize the findings and provide clear recommendations to address the identified problems or maintain correct baseline configuration. Carrying out this audit leaves a clear lesson:

When carrying out the audit, a systematic approach should be followed, checklists can be used if necessary and findings clearly documented. The audit should be performed by people with technical and SCM knowledge to ensure an accurate assessment.

Chapter 5: Status accounting

General guidelines

- Time and budget are expressed in percentage reference values.

- The effort is expressed according to the technical complexity of the tasks as follows:

Low: Routine or well-defined tasks.

Medium: Moderate challenges requiring analysis and design.

Hard: Complex tasks that may involve research and creative solutions.

Too hard: Very complex challenges, possibly without clear solutions initially.

- Human resources are expressed in terms of team skills as follows:

Junior: Team members in the early stages of their careers, requiring supervision and training.

Semi Senior: Intermediate experience, can handle moderately complex tasks with some autonomy.

Senior: Experienced professionals, capable of addressing complex challenges and leading teams.

- The scales are presented taking into account the main stages of the life cycle of a software development project: planning, design and implementation, testing and maintenance.

Time scales, budget, effort and human resources

TIME

Planning: 10-15%

Design and Implementation: 50-60%

Testing: 15-20%

Maintenance:5-10%

BUDGET

Planning: 5-10%

Design and Implementation: 50-60%

Testing: 15-20%

Maintenance:5-10%

EFFORT

Planning: Low to Medium. In this stage, tasks such as defining requirements and planning resources are performed.

Design and Implementation: Medium to Hard. Code development, architecture design and functionality creation.

Tests: Medium to Hard. Creation of test cases, execution and error correction.

Maintenance: Low to Medium. Updates, bug fixes and improvements.

HUMAN RESOURCES

Planning: Senior to lead planning. Semi Senior and Junior to assist with specific tasks.

Design and Implementation: Mix of Semi Senior and Senior to lead the development.

Junior and Semi Senior for specific tasks and learning.

Tests: Semi Senior and Junior for test execution. Senior for designing testing strategies and problem solving.

Maintenance: Senior and Semi Senior to lead maintenance. Junior and Semi Senior for specific tasks and assistance.

Scope of software configuration status accounting

Below is a compliance report on the status of accounting with respect to the previously established policies, taking into account time, budget, effort and human resources of the APPROVED change requests.

This table contains the calculation of budget, time, effort and human resources of the approved change requests, as well as their totals.

C.R.	Budget	Time	HR	Team Skills (HR)	Effort	Effort according to complexity
CR1	1.77%	2.36%	1	Senior	5.2%	Medium
CR3	13.13%	5.83%	3	1 Semi senior, 1 senior, 1 junior	9.8%	Hard
Total	14.9%	8.19%	4	2 senior, 1 semi-senior, 1 junior	15%	Medium to Hard

Taking into account that the calculation of effort, time, human resources and budget of the approved requests (CR1 and CR3) correspond to the Design and Implementation stage, to compare whether the values are appropriate with respect to the policy, the scales of the 4 indicators in the Design and

Implementation stage. Therefore, the following table shows whether the indicators are appropriate taking into account the defined policy.

Design and implementation	Policy	Calculated values	Appropriate
Budget	50-60%	14.9%	Yes
Time	50-60%	8.9%	Yes
HR	Semi senior - Senior	Semi senior – Senior – Junior	Yes
Effort	Medium to hard	Medium to hard	Yes

According to the values presented, the effort, time, budget and human resources required for the approved change requests are in accordance with the defined policy.

Conclusions

In conclusion, for a software development project, software configuration management has proven to be essential to efficiently evaluate and manage change requests, allowing a comprehensive analysis of to what extent the identified configuration elements are affected and in the In the particular case of this project, the analysis was extended to team effort, budget, time and human resources. Implementing rigorous status accounting provides a detailed view of the project status, facilitating informed decision making and ensuring effective control. This approach demonstrated that, if properly applied, it strengthens the team's ability to adapt to changes, optimize resources, and maintain software integrity throughout the project life cycle.

In the specific case of this project, relevant lessons can be highlighted, such as having an adequate decision-making policy that will influence whether the approval of change requests will be done in accordance with what really aligns with the project's objectives and what that it adjusts to the available resources, that updating the project baselines after an approved change request guarantees that at the time of design and implementation the documentation is updated so that no time is lost between the implementation of an initial functionality and a new.