# **Abstract**

“Creativity is essential to computer science students, and computer science makes it easy to be creative” (Romeike, 2007). Creativity, creative design capability, multidisciplinarity, collaborative ability, and artistry can improve computer scientists and software engineers’ abilities in problem-solving, innovation, software design, and development. Music technology is a domain that can be used as an efficient tool for creative development. It is engaging, and “it can help develop creative thought in an academic environment and allow students to gain self-efficacy in their creative abilities” (Rosen, Schmidt & Kim, 2013). When developing music technology projects, students can easily combine art, science, and technology. Whether it is theoretical research or an applicative project, it naturally requires a merge between artistic and computational paradigms and a combination of several disciplines such as music, art, sound, neuroscience, psychology, sports, education, and gaming. While creating and collaborating, music technology education helps students express their personality, passion for music, and other positive emotions (Brown & Theorell, 2006). The combination of academic studies, positive emotion, and enthusiasm is an integral part of optimal engagement, increasing creativity and innovation.

In this work, we developed a creative education method based on music-tech projects development that uses the Muzilator platform as a creative education tool. The platform is a plugin-based web platform that enables developers to split their project into a set of independent plugins to implement, debug, upload, and share them with the platform’s community.

The goal of this study was to learn which project features and team combination can optimize the students learning outcomes, and help students develop their creativity, innovation, artistry, design capabilities, and collaboration skills.

Our research is based on 75 projects implemented by 183 computer science students that participated in the “Computer Music” class in 2016-2020. The students developed ideas and prototypes (POCs) for innovative research or applicative music-tech projects. They worked in teams in an Agile methodology and developed the projects in three sprints. We divided the projects into five main categories and evaluated the projects’ risk level, creativity, multidisciplinarity, interaction, artistry, and creative design. We examined the difference between theoretical research projects and applicative or entrepreneurial projects and analyzed the students’ self-evaluation as well as a subjective report on the final project.

The analysis results show that high-risk projects were more creative and artistic than low-risk projects. Students who considered themselves as self-learners combined more disciplines in their projects than others. Mixed teams (men and women) developed the most creative, artistic, and multidisciplinary projects, while other team combinations were less effective. Soloists (teams with only one member) have shown the lowest rankings in all parameters and learning outcomes. Women tend to choose to develop interactive applications, while men tend to choose more theoretic (algorithmic) non-interactive research projects. Finally, teams that used the Muzilator platform developed projects that were more creative, multidisciplinary, artistic, and were ranked higher in creative design than projects that did not use the platform.

During the writing of this work, some of its conclusions and work process were presented at the following conferences:

1. The 8th Kinneret Conference for Software Engineering Education, February 2020, Israel.
2. The 4th MIC (Marconi Institute for Creativity) Conference – Nurturing Creative Potential (ISSCI), September 2020, Italy.

# **1 Introduction**

Traditional computer science education (academic and non-academic) combines mathematical background, theoretical computer science, computational thinking, computer programming, and software engineering. While those skills are necessary for algorithm design and implementation, additional skills and practices are essential for computer scientists and software engineers to be able to solve complex problems and to innovate:

1. *"Creativity is the use of original ideas to create something new and effective" (Runco & Jaeger, 2012)*. Creative thinking and creative design in software engineering are the sources to improvise and suggest solutions to dominating complex systems.
2. *Multidisciplinary* is the ability to merge between different disciplines to explore and suggest a solution.
3. *Collaborative ability* is an essential skill for computer scientists. Collaborative ability is essential for communication and synchronization between individuals and teams. It contributes to code sharing, the quality of the products, accelerates coding and integration processes, and improves software design capability, testing, and QA. The importance of a software product's design and development is *"dependent on the team members' openness, analyzing a system design, and coding the various components"* (Nelson, Brummel, Grove, Jorgenson, Sen & Gamble, 2010).
4. *Software Design Capability* - the use of software designs while developing is necessary for the project's future maintenance. The developer needs to have the ability to develop independent components as part of a large project, relate, and interact with other system components while having a deep understanding of the global scope.

Music technology is a domain that can be used as an excellent tool for creative development (Rosen, Schmidt & Kim, 2013). A high level of engagement has been shown among students who studied and developed musical projects, and the students were intellectually involved in the process of meaningful exploration (Newmann, Wehlage & Lanborn, 1992). When creating and collaborating, music technology becomes a tool that expresses positive emotions while learning. The combination of academic studies and positive emotion is an integral part of optimal engagement (Khairuddin & Hashim, 2008). Music technology is a multidisciplinary domain that naturally merges between artistic and computational disciplines. When students develop a music technology project, they need to express their software design capability skills and build and combine different artistic or computational components, such as an interface to trace over interactions, a synthesizer, an algorithm, and more.

In this work, we investigated and learned the characteristics of music technology projects and the key factors that can improve computer science students' skills, such as creativity, artistry, multidisciplinarity, creative design capability, and some aspects of software design and collaboration skills.

Our research is based on 75 projects developed by 183 computer science students (3rd-year undergraduates or masters) who participated in the "Computer Music" class between 2016 and 2020. In this class, the students were given an initial background on music technology and learned how to use tools to develop an innovative idea. The project's idea could be either a theoretical research project like an analysis, a generation algorithm, or an applicative project, an intelligent interface, or a POC (Proof of Concept) for a new application. The students worked in teams of one to four members in an Agile methodology. We divided the projects into five main categories and evaluated the projects in several criteria: creativity, artistry, interactivity, multidisciplinarity, and risk. We examined the teams in terms of team size, mix of man and woman, team members' skills and background (in software development and their musical background), and self-evaluation as creative, multidisciplinary, and self-learners.

In 2020, the last year of this experiment, we launched and tested Muzilator, a plugin-based web platform for sharing and collaboration. The platform is an innovative education and collaboration tool and environment for all developers, projects, and teams. We examined the efficiency of working with the platform, its abilities, and how it can enhance the students' creativity, multidisciplinarity, creative design, software design capability, and collaboration.

The document is organized as follows: Section 2 includes background and related work. Section 3 begins with a categorization of "Computer Music" projects, the characteristics we examined on individuals, teams, and projects. In section 4, we describe the Muzilator experiment from 2020. Section 5 presents several computational analysis methods. Section 6 contains our main conclusions and suggestions for future directions.

# **2 Related work**

This section presents a review of related work divided into sections by subjects related to this study: creativity, creative education, project-based learning, and music technology education.

**Creativity**

Creativity in computer science (CS) can be presented in two main perspectives: creativity and the person and creativity in the software development process (Romeike, 2007). The first approach is based on motivation among students (Junius, 2015; Bergin & Reilly, 2005) and explains that highly motivated students raise higher creativity performance than others. Romeike describes multiple factors that can raise motivation: a hope to use the software in the future, an open-source community to enhance tracing and reports of other developers and integrate students into teams by their goals and their chance to extend their programming skills by awareness to different concepts. The second approach describes the importance of a multidisciplinary viewpoint and creative processes in software design. When examining a multidisciplinary process over different domains (i.e., art, creativity, and engineering), it may share common attributes (Charyton & Snelbecker, 2007). They conducted a study designed to understand the differences or similarities between these domains among music students and engineering students.

Several psychology approaches have been used to estimate the differences between the groups, such as the creative personality scale, the creative temperature scale (Gough, 1979), the cognitive risk tolerance survey (Snelbecker, McConologue & Feldman, 2001), the harmonic improvisation readiness record (Gordon, 2000 & Kiehn, 2003) and the creativity test (Lawshe & Harris, 1960). The results determine that engineers and musicians are approximately equal in terms of artistic creativity.

The enhancement of creative development among undergraduate computer science students can be conceptual frameworks (Ferreira, 2013). Ferreira presented a conceptual framework that serves students by focusing on programming and iteration (HCI). Seven rules define the platform: 1. Immerse (solution adaptation to a relative problem), 2. Dependencies Recognition, 3. Exploration of complementary paths (elaboration and sharing), 4. Overcome Boundaries (generalization and high-level scenarios), 5. Expansion or combination of ideas, 6. Discover Unpredictable Places (transform ideas to novelty interpretation), 7. Developing. The results show that Ferreira's framework can let the students enhance their creative thinking, strategies, and programming skills.

Estimating creativity using science and art domains has become popular over the years. Agnoli, Corazza & Runco (2016) defined this problem as multidimensional because it can be tested in several aspects (convergence, divergent, psychology, and more). They presented the Battery test to assess creativity and measurement of ideation and evaluation.

The test includes six steps: 1. Remote Associates Test (RAT) - check relatives between cue words, 2. Title task - suggest alternative titles for classic books or movies, 3. Figure task (Wallach & Kogan, 1965), 4. Exploration of realistic problems instead of abstract problems, 5. Creative Achievement Questionnaire (Carson, Peterson, & Higgins, 2005) - measures creative accomplishments in 10 different domains. 6. Creative Activity and Accomplishment Checklist (CAAC) - participants ranked creativity achievements in several domains. Other tests from psychology have been taken, such as Big-5 and Raven's Advanced Progressive Matrices (APM). The participants (over 300 students from the University of Bologna) took several tests to justify the Battery test. They found that divergent thinking abilities were positively associated with personality traits and with creative artistic achievement. Also, they describe that low levels of ability to solve problems are essential to predict creative achievement.

Nilsson (2011) suggested a methodology to measure innovation and creative designs, the taxonomy of creative design (see figure 1). He presented five hierarchical levels of creative design: 1. Imitation - "is the creation the same as something that already exists?", 2. Variation - "is it a slight change to an existing object?", 3. Combination - "is it a mixture of two or more things such that it can be said to be both?", 4. Transformation - "is it a re-creation of something in a new context?", 5. Original Creation - "does it appear to have no discernible qualities of pre-existing objects?".

Novelty, by Nilsson, is the taxonomy level of being novel, new, or unique and scaled by the taxonomy. It can be measured as two-dimensional parameters: Novelty in Form and Novelty in Content. This taxonomy can be applied to creativity in non-relative fields by scale adaptation, for example: measure creativity in education to determine novelty among students (Junius, 2015).

**Figure 1**: *The Taxonomy of Creative Design (Nilsson, 2011)*

**Creative Education**

Creative educational methods are relevant and necessary in the aim to enhance creativity among students. Rauth, Köppen, Jobst & Meinel (2010) presented an educational method to enhance the creative confidence level. They collected observations by interviews from institutes that are engaged in creative educational thinking.

The interview contained various questions regarding creative education design, and it was based on creative education design (Lande & Leifer, 2010). At the beginning of the experiment, they let the participants become aware of the process and creative assignments and challenges to ensure they understood it. They found out that the participants suggested creative challenges independently, without any relative background to predefined creative challenges. The participants were aware of the uncertainty (risk) level of a given creative assignment.

Nelson, Brummel, Grove, Jorgenson, Sen & Gamble (2010) proposed the SEREBRO (Software Engineering REwards for BRainstorming Online) system for modeling creativity for a computer program. The system provides measurement opportunities to develop metrics of originality, elaboration, and overall creativity. Students worked in teams ("even when a single member is more creative or has an advanced skill set, the success of the project requires the contribution of all members, especially within a small team") and rated the projects by fluency, flexibility, originality, elaboration, and overall creativity. SEREBRO platform assigns reward points to each individual or team for their creative input; For example, the platform methodology rate usages with maximum K points (where K is a natural number). Each usage of a specific process grants the developer K points and 0.5K points to who is used in the process. This way, they measure each team's total score by usage, re-use, and share. The results of this research were precise.

Creativity ranged from 3.18 to 4.84, and in general, teams with higher quality ratings received high creativity ratings. Nelson's work's primary purpose is creativity assessment and how to enhance creativity while developing a system.

**Project-Based Learning**

Project-based learning (PBL) involves solving a given problem in educational activities, and the result is a complete product (Adderley, Ashwin, Bradbury, Freeman, Goodlad, Greene, & Uren, 1975). To understand the effect of PBL on students' creative thinking, Mihardi et al. (2013) used the KWL worksheet, a framework that connects prior knowledge of a student to actively learning (Ogle, 1986). They selected students with random sampling to participate in the experiment (Fraenkel, Wallen, & Hyun, 1993). The participants were asked to implement and solve a factory design problem and fill pre- and post-questionnaires. The results have shown that students' creative thinking in PBL is higher than in other methods.

Furthermore, PBL enables the students to propose groups’ ideas to reach their final goal, an end-to-end project. PBL is considered a suitable method for preparing students with the expected skills for group creativity. Zhou et al. (2009) tested group creativity in the development of PBL among engineering students. The participants were two groups of master engineering students, and they collected data by observations during the experiment.

The groups were asked to complete a mission from the engineering field and deliver a report. The research found that peer learning (learning by other group members) is oriented by project type and field. Besides, they found that PBL can build wild knowledge for students. When we drill down, these conclusions are the influences of PBL on students learning and learning by collaborative behavior.

**Music Technology Education**

The maturation process of the New Interfaces for Musical Expression (NIME) field has a growing interest in teaching the design and implementation of Digital Music Instruments (DMIs) and finding objective evaluation methods to assess the suitability of these outcomes. Jorda & Mealla (2014) proposed a methodology for teaching NIME design with a set of tools meant to inform the design process. This approach has been applied in a master course focused on exploring expressiveness and the role of the mapping component in the NIME creation chain through a hands-on and self-reflective approach based on a restrictive setup consisting of smart-phones and the PD (Pure Data) programming language. The outcomes of the students gained through this iterative methodology were: 1. All of them (some of whom had never performed music or programmed computers) were able to effectively engage in the NIME design processes, developing working NIME prototypes that fulfilled all the requirements; 2. The assessment tools proved to be a consistent method for the evaluation of NIMEs systems and performances;

3. Informing the design processes with the outcome of the evaluation showed traceable progress in the students' outcomes. Although these findings were obtained in the specific context of a NIME course, they believe that several of these solutions and learnings could be extrapolated to more generic contexts, being other NIME or even HCI courses, design methodologies, and evaluation methods for both fields, and could therefore inform teachers, designers and practitioners in general.

The design and development of musical controllers among musicians and novices were presented at New York University (NYU) by D'Arcangelo (2002). While no formal musical background was required for the class, musicality was positioned as the design process is a driving force. The class was an experiment with an educational approach that required each inventor to set his/her musicality standards, although nascent, as the basis for musical interface innovation. The design challenge was articulating expressive goals based on these musical standards and then working back to the tools and technologies required to achieve them. Early discussions on the qualities of music and what constitutes musical expression helped students to articulate the musical direction they chose to pursue. Their notion of music was open and egalitarian. The class encouraged sensitivity to how musical styles vary across cultures, throughout history, from the sacred to the profane, within popular and classical settings, and with the advent of new technologies.

However flexible and open their definition of music was, each student needed to adopt some sense of musical style to root the invention of his/their new instrument. As a result, the project was developed explicitly to break from the traditional musical instruments' paradigm and present new musical expression models.

A framework that enables a quick design and prototyping of passive mobile device augmentations was introduced by Michon et al. (2017). This framework is suitable for developers with a background in music, sound design, and FAUST programming language for synthesis. They organized a one-week workshop at Stanford's Center for Computer Research in Music and Acoustics (CCRMA) and taught seven participants how to make basic musical smart-phone apps using their Smart Keyboard App Generator. Besides, they taught them how to use 3D printing for mobile device augmentations that enable users to make sounds or even use the phone as an instrument. The participants were free to invent, design, and make any musical instrument or sound toy for their final project. In one week, participants mastered all these techniques and designed and implemented very original instrumental ideas.

# **3 Computer Music Education for Skills Development**

This section describes our educational method, categorization of music-tech projects, and analysis results of project evaluation. We examined 75 music-tech projects developed by 183 students between 2016 and 2020.

The main questions we asked are:

**RQ1**: In what ways do students' and teams' characteristics align with the project's creativity, multidisciplinarity, artistry, and risk level?

**RQ2**: In what ways does the music-tech project type align with the project's quality and students learning outcomes?

**RQ3**: How can a team composition affect the project’s quality?

**RQ4**: What music-tech characteristics are interdependent and affect the creativity level of the project?

We begin in section 3.1 with a characterization of music-tech projects. Section 3.2 includes definitions of students' skills and project characteristics. In section 3.3, we describe our experiment method and students' and projects' grading. In section 3.4, we present the evaluation analysis results.

## **3.1 Computer Music Projects Categorization**

First, we propose a categorization of computer music projects. The five categories are essential to artistic and computational models that can be used to develop a project in the category. We divided all 75 projects according to these categories and analyzed the projects’ characteristics in different categories.

1. **Music experience** - An application with specific music functionality (i.e., playing or learning), which does not involve creation or generation. This category includes musical games that combine musical elements, sounds or musical pieces, educational applications, players, streamers, recorders, editors, or digital controllers. Applications in this category are interactive applications, where the interaction is more functional rather than artistic, but the application does not make artistic decisions. Although some of the application details may change during the development process, the developers can plan and design the project in detail before the development process. In terms of how valuable the user experience is for the user, the quality of the product is not guaranteed. Nevertheless, the application can be defined and fully illustrated and planned, making the level of risk relatively low.
2. **Creative expression** - An interactive application that displays a musical interface to the user and can respond to his interaction. In this category, applications take artistic considerations on the interaction with the user. For example, an application for music creation in which the user creates.
3. **Analysis and Generation** - An algorithm that analyzes or generates music pieces like a MIR (Music Information Retrieval) algorithm for feature extraction of genre classification, a personalized playlist generator, or a generation algorithm (music, visualization, etc.). A generation project is not interactive.
4. **Smart Interaction** - An application that combines user interaction and creative expression with analysis or generation. For example, an application that analyzes users’ interaction or music improvised by the user generates a response that is played to the user.
5. **Sonification** - a data-driven project that uses non-speech audio to convey information or perceptualize data. A sonification algorithm builds an auditory representation for given data, such as sonification of stock prices, a text, or brain activity. Sonification can be used for scientific, experimental, or artistic purposes. In such a project, the developer may have a general idea of how he/she wants to convert the data, but most details are determined during the development process when the data is processed, and the developer is more familiar with it. One can say it is a generation project, but here is an additional level of abstraction.

In sonification projects, data is first transformed from another domain, and then the generation is done, which is different from generating music using compositional rules or music pieces. We consider such projects as high-risk projects.

## **3.2 Characteristics of Students and Projects**

This section includes students' and projects' characteristics we have investigated in this work:

1. *Artistic ability* includes skills and talent to create delicate works of art: painting, drawing, sculpting, musical composition, etc.

An *artistic application* is an application where the students use or combine musical elements or artistic aspects in their project.

For example, an application that interacts with a human enables him to create a piece of art using an algorithm that analyzes or generates music.

1. An *Interactive application* is an application that allows users to enter data or commands, like a controller or instrument, a music player, a synthesizer, an educational application, a DAW (Digital Audio Workstation), an application for music creation, etc.
2. An *Artistic-Interactive* application is both an artistic and interactive application; for example, an application that gets music improvisation from the user interacting and analyzing, generating, and playing a musical response. A music player is an interactive application since the user interacts with the application by entering functional commands like: 'play', 'stop', 'like', but it is not an artistic application, and therefore, it is not an artistic-interactive application. Another example is an application that analyzes musical pieces and generates a new musical piece based on the other pieces. It is an artistic application, but it is not interactive.
3. A *multidisciplinary skill* can combine multiple disciplines to redefine problems outside of normal boundaries and reach solutions based on a new understanding of complex situations.

A *multidisciplinary project* is a project where a few disciplines are incorporated into the project to solve it.

1. *Creativity* is the skill or talent to use imagination to create and solve a problem. A creative project is a project where a relatively high level of imagination and originality was used to solve the problem and create an original, unique, and innovative project. A creative project is not necessarily an artistic project. For example, unlike any other game, a new game is a creative project, but a music player is not a creative project since it imitates standard techniques and interfaces for music playing.
2. *Elaboration* - The ability to elaborate part of the project (component), engage, describe the number of dependent components, and the ability to isolate components.
3. The *novelty in form* and *novelty in content –* according to *Nillson's creative design taxonomy, 2011)*: a two-dimensional creativity assessment model, from complete imitation to original creation (originality). The dimensions scaled 1 to 5 (imitation, variation, combination, transformation, and original creation) and is interpreted by:
* The *novelty in form* - Describes the novelty in the project source code: how many new components, different architecture (according to the initial project given in class, and the assignment was based upon).
* The *novelty in content* - Describes the novelty in the project content: algorithms, out-source libraries, complexity run time, optimizations.
1. A project with a *high level of risk* is a project where the main idea and the problem it is aimed to solve can be defined clearly before the development begins, but many designs and implementation details are unknown or unclear in advance. Some research and trial and error process are needed to define them and move from one phase of the development process to another. Therefore, it is uncertain on the project outcomes, and whether the students will succeed in achieving their goal and solve the problem, they are aimed to solve.
2. Entrepreneurial project vs. research project
* An entrepreneurial project is a project where the students have an idea for a product that solves a problem. The project outcome is an application prototype, a POC that will be developed and tested on potential users.
* A research project is a scientific endeavor to answer a research question. Specifically, projects may take the form of case series, case-control study, cohort study, randomized, controlled trial, or secondary data analysis such as decision analysis or meta-analysis. Besides, the students have some questions they wish to answer, and they develop an algorithm or an application to try to answer their questions.

## **3.3 Method**

### **3.3.1 Educational method**

The class started with introducing music technology, followed by a discussion on current needs and future directions in this domain. In the following three weeks, the participants learned about musical elements in theory and practiced it using the SuperCollider language: notes, pitch, timbre, tempo, rhythm, melody, harmony, texture, structure, and the MIDI protocol. After four weeks, the participants were divided into teams of one to four participants. They were asked to present and discuss three ideas for the final project in class and choose one out of the three. The following task was to build a presentation that describes the project (see appendix 5). This presentation was updated after each phase and used in the final presentation of the course demo day.

During the development process, each team had two meetings with the course teachers: The first meeting took place after the team created three ideas for their project. The second meeting occurred after the first development phase (or sprint).

In the first meeting, the team members presented themselves, their background, interests, and three ideas and possible solutions in addition to the idea and development options. We discussed each idea's level of risk and how it matches the team members’ interests and abilities.

After choosing one of the three ideas for the project, the students started planning an Agile development process and dividing the development process into three steps. At the end of the step, a deliverable and working part of the project will be delivered and tested with the potential users. In the case of a non-interactive project, the deliverable would be a preliminary output of the algorithm. Some audio output examples or videos that demonstrated the user using the prototype should be submitted in both cases. The teams had to learn the problem domain and solve it in a learning-by-doing or PBL process. Each team had to review relevant papers, choose one paper that is most relevant to their project, and deals with the same problem their project is aimed to solve, or a similar problem. They presented the paper in class, followed by their project presentation that describes their project goals and the division into three steps. A discussion and feedback were given in class. A demo day took place after the end of the semester, and the students presented their presentations and projects.

### **3.3.2 Project evaluation method**

To learn about the projects’ characteristics, the course teachers and the students ranked the projects in 2020. We collected data from the students at the beginning and the end of the semester via pre- and post-questionnaires. At the beginning of the semester, the students were asked to rank their creativity, multidisciplinarity, and autodidact abilities and provide information about their music and software programming background.

 **Table 1**: *The participants' pre-questionnaire*

At the end of the semester, they were asked to rank their projects and their learning outcomes. The students were ranked according to the following criteria from the data we collected in the pre-questionnaire (see table 1).

The grading method for Musical background:

* + - **1** - No background.
		- **2** - Beginner - Played an instrument for 1-2 years.
		- **3** - Intermediate - Played an instrument for 3-5 years.
		- **4** - Advanced - Played an instrument for at least five years, played additional instruments or sang, or majored in high school music or conservatory music.
		- **5** - Expert - Academic background in music or a professional musician.

The grading method for Professional background:

* + - **1** - No experience.
		- **2** - Trainee - 1-2 years of experience in a student position, a technological position other than a developer in the IDF.
		- **3** - Junior - 1-2 years of experience as a senior developer in the industry (or IDF).
		- **4** - Senior - 3-5 years of experience as a programmer in the industry (or IDF).
		- **5** - Guru - More than five years of experience in the industry (or IDF) and additional experience as a team leader or specific expertise in machine learning, data science, backend, etc.

The projects were ranked according to the subjective criteria (see table 2).

**Table 2**: *The projects' evaluation criteria and scale*

## **3.4 Main Results**

We present the students' projects’ distribution according to project categories (see figure 2). 67.6% of the projects were interactive projects from the music experience and creative expression categories, while 13.5% of them are analysis and generation, and 10.5% of projects combine interaction and algorithms.

**Figure 2**: *The distribution of students’ projects by category*

The following is a summary of the main results. First, we refer to the student's analysis; second, we refer to the team and the project developed by the team.

### **3.4.1 Individual**

Evaluating or measuring creativity is not trivial. To achieve a high quality of the project's creativity ranking, we ranked the projects in two ways: 1. Following definition 5 in section 3.2, we ranked a projects' creativity level by assessing the projects' idea and implementation’s overall imagination and originality. 2. We used Nilsson's taxonomy for creative design (Nilsson, 2011) and ranked each project according to his model's two-dimensions: novelty in form and novelty in content. Table 3 includes the projects' average grade of each category. The comparison (see figure 3) shows a high correspondence between the two ranking methods.

**Table 3**: *Creativity average and standard deviation of projects by category*

**Figure 3:** *A correspondence between Nilsson’s creative design and the average creativity rank*

Following are conclusions on the individual level:

1. Participants who defined themselves as autodidacts were more explorative and combined more new disciplines in their projects. Their projects’ multidisciplinarity and artistic rates were relatively higher than in other projects.
2. Participants who ranked themselves as highly creative developed a project with a higher creativity rank.
3. Participants who developed projects with the highest level of risk have high self-esteem in all factors as autodidact, creative, and multidisciplinarity.

**Table 4**: *Participants’ self-esteem characteristics average compared by projects’ risk level*

1. No significant difference was found between men's and women's self-esteem as an autodidact, creative, and multidisciplinary. Nevertheless, one can see that men's rates are slightly higher than women's in all categories.

**Table 5**: *Participants’ self-esteem characteristics average compared by gender*

### **3.4.2 Project**

1. High-risk projects are more artistic and creative than other projects, and vice versa (RQ1).
2. Teams who developed a project with a low level of risk raise got lower creativity rates (in both the creativity ranking methods). On the other hand, participants who developed a high level of risk projects raise higher creativity rates (RQ1).
3. A strongly positive correlation (=0.876) was found between the projects' creativity and multidisciplinarity. Students who combine more disciplines in their projects tend to be more creative students (RQ1).
4. Students who developed projects with high risk developed more creative projects and combined more disciplines in their projects. Further to the previous conclusions that creativity and multidisciplinarity have a strong dependency that can affect how the project developed, we compared those variables and risk levels by project type (see table 6). This analysis reinforced the conclusion that project type can affect the students’ creativity. For example, students who developed Sonification and Generation projects raise high multidisciplinarity and creativity rates (RQ2).

 **Table 6**: *Creativity average and standard deviation of projects by the participants' characteristics*

# **4 A Collaborative Plugin-Based Platform as a Creative Education Tool**

This chapter describes our main experiment on the Muzilator platform as a creative education tool to enhance creativity, artistry, multidisciplinarity, and collaboration skills. This experiment was the first pilot done with the platform on a relatively large group of users, 47 Computer Music students (32 men and 15 women) who took our class in 2020. Our goal was to learn about the platform's contribution to the students’ and the temas’ learning experience and outcomes, and the projects' creativity and quality.

## **4.1 About Muzilator**

Muzilator is a plugin-based web platform for interactive applications intended for musicians, novices, developers, and researchers (Hollander-Shabtai & Peretz, 2020). For app users, Muzilator improves creative musical expression, interaction, and creative skills by interacting with Muzilator’s interactive musical interface and applications. For developers, Muzilator exposes APIs that can easily add their plugins to the platform. Muzilator records all interaction data and data transferred between plugins, enabling researchers to build or use existing plugins or apps in their experiments and analyze the recorded data. The Muzilator platform is designed in a plugin manner (see figure 4), and a plugin may have any functionality.

**Figure 4**: *Muzilator hosts web applications as plugins*

There are two main types of plugins (see figure 5): Applications (App) and Libraries (Lib). An App can be, for example, an interactive musical instrument, creation or educational app, or a game. Libs can be a controller, an external MIDI device, an analyzer (online/offline), a sound engine, a profiler, etc.

**Figure 5**: *Muzilator plugin types: Apps and Libs*

All plugins can communicate one to each other with Muzilator channels (see figure 6). The channels transfer data from plugin A to plugin B if a channel is defined between those plugins. Each Muzilator App can use any number of Libs. The App is responsible for: loading Libs, connecting channels between plugins, and for the App logic that uses and synchronizes between the Libs. The Muzilator architecture design allows any web application to be uploaded to the platform as an independent App or Lib. Each plugin can be developed by a different developer and can be integrated easily with other plugins. Muzilator developers can benefit from being a part of a community of developers that create interactive musical applications and share any part of them with the community as plugins.

**Figure 6**: *An instrument App uses two Lib plugins: A controller and a sound engine.*

As an initial set of plugins, we designed and developed a set of plugins and tools (integrated to Muzilator) for all students, such as a dedicated debugger, which helps with communication between plugins, and as a tool for students with no musical background. We created tutorials and guides that were handled to the students with the basic set of plugins.

## **4.2 Creative Education and Collaboration Tool**

The uniqueness of the educational method with the Muzilator platform can be reflected in several aspects:

* + - 1. *Development of independent shareable plugins* - The students develop their ability to focus on a specific entity as a plugin to write their plugin or use independent entities that already exist in our platform is a vital software design capability skill. From our experience, without this mechanism, most students failed to separate between different components or layers of the projects, which resulted in lousy coding and complex development or maintenance processes. Also, the students can focus on creative ideas on the responsibility of a specific plugin and optimize its functionality and uniqueness.
			2. *A platform for everyone* - The platform architecture enables students with or in any level of programming skills with or without musical or artistic background to write plugins and easily add them to the platform. The plugins are written in JavaScript, a widely common programming language for web development and the applications are browser-based applications that can use the Web-Audio API that is commonly used today. It enables the students to focus on the innovative and musical aspects of their project.
			3. *Use of existing plugins* - The students have a variety of artistic and computational projects. The participants' choice to develop was based on their preferences: artistic HCIs, sound-engines, players, recorders, profilers, applications, online/offline algorithms for analysis, music generation, prediction, profiling, and more.
			4. *Software design and software engineering* - The platform exposes an SDK to build and integrate plugins quickly and easily (see appendix 2). The SDK can be used in any JavaScript framework and installed via any web packages installer. The platform also suggests a state machine interface that conveniently presents a state machine’s concept and its use (see appendix 1). Developers and students can share their applications’ ERD (Entity Relationship Diagram) with the community for future use.
			5. *Work with a community* – Working as part of the platform community of at least 47 participants enabled the students to achieve a comprehensive perspective of the platform design, components, and experience integration processes, to collaborate with individuals and other teams who work on other projects and share their plugins.
			6. *A unique Agile and artistic process* - requires the students to develop a project in three sprints, share the project deliverables at the end of each sprint, use other projects, and give other students feedback. This process was guided and monitored by the course team.
			7. *Write and use APIs* - The ability to combine and communicate between independent web applications through a unique API. The students learn how to bind an out-source platform, write their own plugins API, and expose it to the community, and use an API of other shared plugins.
			8. *Versatility* – Since a plugin can have any functionality in any domain, the students can easily combine art, technology, and science between different disciplines.
			9. *Data recording and storage* - The platform has a built-in data storage mechanism (i.e., the recorder) information for interaction between users to perform analysis and optimization for development processes.

##  **Experiment and Educational Method**

The educational method combines a learning process divided into three phases and uses a plugin-based platform for musical applications. The three phases are Assignment 1 – An HCI, assignment 2 – A computational plugin, and the course final project. In the first two phases, the students focused on exploring a specific interactive musical application component. They received an initial plugin project and continued to develop it independently. The submitted project was uploaded to the Muzilator platform. In the following, we describe each of the three phases:

* + - * **Assignment 1** - Exploring an HCI - designing an interface plugin - In this assignment, the students focused on user interaction and the user interface of a simple musical application (controller). Using creation methods from music and art, we provided the students a or theme or a trigger for a new idea. In this experiment, we used "Bubbles", a simple and basic controller that displays random circles with random colors (see appendix 3). Each circle is mapped to a random note (see figure 7a). The students were asked to develop a music controller or a simple musical application for a specific purpose: music interaction, a game, or a tool. They designed and implemented the controller's display while considering the target user's interaction and experience.

The students had to combine programming and artistic abilities and design the HCI's features like size, color, shape, configuration (spatial organization), graphics, animation, movement, gestures, mapping of graphical elements to musical elements, musical context, human factors, and use of photos and videos. They also had to adjust some of the features to the potential user to optimize his interaction. In addition, the students were responsible to send the user's interaction data from their plugin to other plugins that will use it in the future. Figures 7, 8, 9 demonstrate examples of the students' assignment 1 with three levels of abstraction: Figure 7.b-f demonstrates five different uses with minor graphical changes (mostly in shape, configuration, colors, spatial organization, and pitch mapping to a circle). While figures 7b and 7c show a simple controller where a significant focus was given to its design, the spatial organization that considers relations between notes and chords, figures 7d, 7e, and 7f that show an eye tracker controller where the user plays a melody using his eye movements, and the primary focus was on human factors. Three different configurations were designed and used for three scenarios and musical contexts. Figure 8 demonstrates the next level of abstraction of Bubbles. In these projects, the students designed a tool or a game with a higher level of sophistication. Additional elements were combined in the interface and added some logic, animation to an interface for a tool or a game.

**Figure 7**: *Controllers developed by the participants in assignment 1 - Visual Transformation.*  *Figure 7a is the given “Bubble” controller. Figures 7b and 7c demonstrate simple musical controllers with a specific design for specific musical elements, and figures 7d, 7e, and 7f demonstrate three different configurations for an eye-tracking musical controller.*

**Figure 8**: *Controllers developed by the participants in assignment 1 – Music Composition. Figure 8a is a simple app for music composition where the user composes a melody, and the app continues his composition. Figure 8b shows a variation of a Word-search game. A searched word are triplets of colors of flags. When the user clicks on a circle, a part of a national anthem is played. The user has to find a triplet where the same anthem is played and then choose the right flag. Figure 8c demonstrates an application that learned how the user perceives a melody in a 2D space. The user plays a melody in an empty canvas and the application several times, and the application generates a controller for him. Figure 8d demonstrates an animated chords-game where the user creates a chord by choosing three notes. The notes are mapped to the animated circle that moves in the black rectangle. Each time a circle hits one of the edges of the rectangle, a note is played.*

**Figure 9**: *Controllers developed by the participants in assignment 1 - Generalization. Figure 9a - soundman - a musical Pacman game where the user navigated with sound. Figure 9b shows a musical snake game, and Figure 9c shows the Bubbles controller converted to a 3D VR controller with additional abilities such as drag and drop that enables the user to organize the elements in a 3D space.*

**Figure 10**: *Sonification projects developed by the participants. Figure 10a shows stocks graph sonification. Figure 10b shows a musical paint app where the user draws a painting, and the application plays the sonified painting. Figure 10c shows an application that takes short stories, and using sentiment analysis and sonification, creates a playback for the reader while he reads the story.*

* + - * **Assignment 2** - A computational Plugin. In this phase, the students focused on a logical component of a musical application, such as an analyzer that analyzes the user interaction and responds accordingly (see figure 11). The students were asked to build a computational plugin for another student's HCI. With this, they gained the ability to be part of a developers' community while also learning the importance of collaboration. Computational plugins can be performed in several approaches, such as a generative algorithm that generates music using computational models (Markov chains, genetic algorithms, google magenta, etc.), an analysis of music played by the user in order to decide whether to switch state in state-machine or analysis of user input in a game and calculation of the score.

 **Figure 11**: *An example of three applications that used the same controller with different analyzers. The first application used the Markov Chains analyzer, the second application used the Genetic algorithm analyzer, and the last used the State Machine analyzer.*

* + - * **Final project -** The students developed an idea for an original music application and implemented it in the following way:

The students were divided into 19 teams of one to four participants in each team.

Each team designed and developed an original music application, such as an interactive app, instruments, generation algorithms, sound engines, sonification, or a game.

The development process was divided into three sprints (Agile methodology), where at the end of each sprint, the students submitted a deliverable project that could be used and tested by the application’s potential user.

In addition to the code of the project, the participants were asked to submit two additional files:

1. **API** (see table 7) - An application program interface which includes:
	* 1. **Plugin Description** - a description of the plugin functionality and how it works.
		2. **ID** – the plugin unique id for re-use and collaboration as registered in the *Muzilator* platform.
		3. **Messages API** - the type of messages the controller can handle their content, for both input and output messages.
2. **Channel-Diagram** (see figure 12) – a diagram that includes:
	* 1. Plugins scheme.
		2. Active channels.

**Table 7**: *An example API of Muzilator plugin* **Figure 12**: *Example of Channel-Diagram. The controller and the sound engine are communicating and send messages on the midi channel.*

The following is examining the development differences between music technology projects: *Music experience* and *Sonification*.

### **Example 1: The Cross Flags game – a music experience project**

"Cross Flags" is a music experience game that shows a variation of a Wordsearch game, where the searched words are triplets of colors of flags. When the user clicks on a circle, a part of a specific country's national anthem is played. The user must find a triplet where the same anthem is played and then choose the right flag.

The development process was carried out in three phases (sprints):

* + **Phase 1** - Create a touch controller in a fixed size (four rows and four columns) and spread randomly different colors with predefined constraints, such as green-white-blue must appear at least once, etc. At the end of this phase, the controller (HCI) handled user interaction (play sound according to the event), but there is no logic behind it.
	+ **Phase 2** - Design and build a computational plugin that gets as input the user interaction data, analyze the pattern, and searches for predefined sets of colors that create a known flag. At the end of this phase, the application, consisting of controller and analyzer, can identify at least two different countries.
	+ **Phase 3** - Generalization and end to end project. Using the prototype, which is defined in stages 1 and 2, the team were required to generalize the project and make it is scalable, i.e., the size of the game board can be determined by the user, the collection of countries will be increased, the audio option is more in-depth than simple midi sounds and more.

Since it is a known game that was converted to a musical game with simple adaptations, once the game was planned and designed, the team could start developing it and faced mostly technical concerns rather than user experience or another issue, which reduced the risk level made the development process more manageable.

### **Example 2: "Stockify" – a sonification project**

"Stockify" is a sonification project that transforms companies' stocks' trading data into auditory data. The application displays a company list and a calendar to the user, the user selects a company and range of dates, and the application plays those stocks.

The development process was carried out in three phases (sprints):

* + **Phase 1** - Create a controller that displays the companies list, the calendar, and the output chart. The chart is determined by the selected date range and displays the stock chart for that period.
	+ **Phase 2** - Design and build a computational plugin that gets as input a stock sequence and returns the MIDI notes mapping that describes the stocks in an auditory approach.
	+ **Phase 3** - Generalization and end to end project. External APIs have been added to extract information about the companies, stocks, and various dates to create a complete product.

Throughout the process, the team learned the complexity of data transformation. Sonification projects, and data transformation into auditory data in general, can be designed for several goals:

1. **Scientific** - Transforming data into auditory data can be used in data exploration, finding patterns in the data, and more.
2. **Experience** - In contrast to the scientific aspect, the main goal is to experience the data, hear it, and enjoy the musical experience generated from raw data.
3. **Musical** - Projects of this type deal with the data's behavior and their translation into an audio representation to create a melody representing the data.

The challenge of data transformation is the ability to map the data so that the output is melodic, with a musical sequence, and auditory conclusions can be drawn. Usually, this challenge is the most difficult and requires analysis carried out throughout the process to find and define the most appropriate transformation for this problem.

## **4.4 Experiment Results**

The participants could choose whether they want to use the Muzilator platform in their final project development or not. We divided the projects into two groups: Muzilator projects and Independent projects. The following table demonstrates the participants' distribution between the groups:

**Table 8**: *The distribution of the participants and the projects*

The participants of 2020 were asked to fill a post questionnaire. They were asked about teamwork, certainty level of their projects, creativity level, and combination of art, science, and technology.

We compared the participants' answers in their pre- and post-questionnaire results. A negative difference represents a student who defines himself as a high level in any given attribute (compared to the course team or his post-questionnaire). A positive difference represents the opposite, meaning the student managed to produce a higher rated project than his self-rating.

### **Individual**

We compared the participants' self-esteem ("the positive or negative evaluations of the self, as in how we feel about it", Smith, Seger & Mackie, 2007) that was reported at the beginning of the semester and compared the results according to the project that they developed (Muzilator or independent project). The average ranking of all participants who developed Muzilator projects was consistently lower than participants who developed independent projects, as demonstrated in the table below. We found two possible explanation for their choices:

1. Students who develop Muzilator projects were less confident or familiar with other environments or wanted to learn more or use a more structured and dedicated tool.
2. Students who develop independent projects were more confident in developing in an environment that is more familiar to them or did not want to spend more time learning in a new environment. By comparing the participants' self-esteem to their creativity, autodidact, and multidisciplinarity (see table 9), we found that participants who chose to develop their projects independently rated themselves higher than others. However, participants who developed their project using the Muzilator platform rated themselves lower than others.

The participants' self-esteem difference can be due to the participants with professional knowledge, which may rate themselves as highly creative and autodidact and develop their projects according to familiarity in their developmental environment and abilities. Participants who developed their projects in Muzilator had a lower rate of self-esteem. By developing independent plugins to Muzilator, the participants used other participants' plugins dedicated to a specific task or computation used by any platform user. In this case, the participants may feel comfortable using an existing platform with dedicated computational tasks and not to develop their projects independently.

**Table 9**: *A comparison of participants' self-esteem by projects’ category. Muzilator projects were developed by 55.4% men and 44.6% women, and independent projects were developed by 69% men and 31% women.*

**Post-experiment analysis**

1. Most of the participants who developed Muzilator projects and considered themselves highly creative developed more creative projects than participants who developed an independent project.
2. Similarly, we compared the participants' self-esteem on their creativity and the project creativity rate. We found that out of 70% of the participants who used the Muzilator platform rated equal or higher creativity rates in their projects than their self-esteem as creative. However, 73.3% of the participants who developed independent projects rated a lower creativity rate in their project.

**Table 10**: *The difference between pre- and post-questionnaires in creativity*

### **Team / Project**

* + - 1. Muzliator projects got a higher ranking of creativity and multidisciplinarity than Independent projects (see table 11). The table compares Muzilator projects and independent projects that were developed in 2020.

 **Table 11**: *Muzilator projects ranking compared to independent projects*

* + - 1. Elaboration and Nilsson’s taxonomy rates were increased during the experiment milestones (see figure 13).

**Figure 13**: *Elaboration and Nilsson’s taxonomy rates*

### **Gender Differences**

In order to figure out the difference between the rates of men and women, we divided our findings into two sections:

* + **Pre- experiment analysis**

We compared the participants' self-esteem reported at the beginning of the semester and compared men's and women's answers. There is no significant difference in self-esteem between men and women as an autodidact, creative, and multidisciplinary. When considering the comparison between men and women's self-esteem, one can see that women rate themselves as an autodidact, creative, and multidisciplinarity slightly but consistently lower than men ("Female programmers are less confident than male programmers", Kay & Shipman, 2014). Later in this chapter, this measure seems to remain constant throughout various comparisons.

 **Table 12**: *The participants' self-esteem average compared by gender*

* + **Post- experiment analysis**

To explore the difference between women and men deeply, we examined different parameters according to their gender, such as project type (see figure 14), creativity, multidisciplinary, risk, etc.

**Figure 14**: *Projects' category distribution by gender*

Teams that contain a certain percentage of women develop more artistic and interactive projects (see table 13). Teams with women only developed more artistic interaction and interactive projects than other teams, and mixed teams developed more artistic projects than other teams (RQ3).

**Table 13**: *Artistic project, artistic interaction, and interactive levels compared by gender*

Teams with women only developed more artistic projects than other teams, and mixed teams developed more interactive projects than other teams (see figure 15).

**Figure 15**: *A comparison of artistic project, artistic interaction, and interactive levels by team composition*

1. We examined the same aspects in the 2020 experiment. Teams with women only developed more artistic projects and interactions than other teams. The interactive level is almost equal between men and women with a slight tendency to men.

**Table 14**: *Artistic project, artistic interaction, and interactive levels compared by gender mix in Muzilator experiment*

1. Both genders developed more entrepreneurial projects than research projects.

 **Table 15**: *The distribution of research and entrepreneurial projects*

1. Musical background (MB) affects women’s creativity more than man’s creativity. A strong negative correlation was found between creativity and musical background among women (p=-0.64), while the same comparison among men raised a weakly positive correlation (p=0.24). As the musical background is lower among women, they developed more creative projects than men with a low musical background.
2. Professional background (PB) affects women's artistry level (p=0.56). Women with professional backgrounds developed more artistry projects than other women. There is no dependency between the projects' artistry level to their professional background (p=0.06) among men.
3. Men developed a higher level of risk project than women (see figure 16).

 **Figure 16**: *A comparison of projects’ risk level by gender*

### **Muzilator Experiment Results**

The participants could choose whether they want to use the Muzilator platform in their final project development or not. We divided the projects into two groups: Muzilator projects and independent projects. The following table demonstrates the participants' gender distribution between the groups:

 **Table 16**: *The distribution of the participants' gender and projects*

The results show that women developed creative and multidisciplinarity projects more than men in both types of projects (see figure 17).

**Figure 17**: *A comparison of creativity and multidisciplinarity levels by gender*

# **5 Analysis**

This section presents an analysis using several statistical methods to examine the influential characteristics of projects’ creativity among students. Our goal is to find an estimator for creativity given project characteristics. We hope to suggest to the students a specific project type that will encourage their creativity level while working on the project. We analyzed 75 projects developed by students between 2016 and 2020. To analyze our dataset, we define the following variables:

***V*** - a features vector of a given project:

**V** = (*M*, *CMPT*, *R*, *A*, *AIN*, *RE*, *ENT*, *MG*, *FG*),

where:

***M***  - multidisciplinarity level,

***CMPT*** - Computer Music project type,

***R***  - risk level,

***A*** - artistic project level,

***AIN*** - artistic interaction level,

***RE*** - research project indicator,

***ENT*** - entrepreneurial project indicator,

***ME*** - total number of members,

***MJI*** - gender majority indicator (1 - men, 2 - equal, 3 - women).

* + - 1. ***C(V0)*** - the creativity level of *V0*, where *C(V0)* ∊ .
			2. **PrM** - a 75x9 matrix, where the *ith* row represents the projects’ vector *Vi*:
			3. **CrV** - a 75x1 matrix, where the *ith* row represents the projects’ creativity level, ***C(Vi)***.

First, we describe two statistical tests: Kendall Tau-b and Somers’ D test. Then, we present an evaluation of the ordinal classification for creativity assessment and analysis of the results.

## **5.1 Statistical Tests**

We used our ranked projects as classified data and evaluated tests to understand the relationship between the ranked projects and creativity. The performed tests are the Kendall Tau-b test (Kendall, 1938) and Somers’ D test (Somers, 1962). We used those tests because the PrM matrix contains categorical data as well as the CrV matrix.

For each test, we defined the following hypothesis:

Let *Fi*be the *ith* column in *PrM,* where

**H0**: *Fi* and *CrV* are independent (not associated) variables.

**H1**: *Fi* and *CrV* are dependent (associated) variables.

The following sections are the tests results.

### **5.1.1 Kendall’s Tau-b test**

A Kendall's Tau test can be used for hypothesis testing for a small sample size (at least ten independent observations). In our case, the sample size is 75. It is a non-directional test (i.e., for two ordinal variables A and B, the results are the same for A-B and B-A), and we used it to generalize associations between creativity and all other characteristics. Kendall's correlation coefficient (Tb) scaled from -1 to 1, where:

1. Tb = -1 indicates a perfect negative monotonous relation.
2. Tb = 0 indicates no monotonous relation at all.
3. Tb = 1 indicates a perfect positive monotonous relation.

After Kendall's test was performed, we converted the results into a spearman correlation (Walker, 2003). The tests were chosen according to the sample size and the types of variables.

As a result, we found that multidisciplinarity, projects' type, risk level, artistry, and research level scored as the highest Tb correlation coefficient value with creativity .693, .246, .284, .610, and .314, respectively (see table 17).

Those features also raise a p-value significance lower than 5%, and by that, they reject H0 and accept H1 with a 95% confidence level. We converted Kendall's tau-b coefficients to spearman coefficients to strengthen the claim, which justified the strong dependency with creativity. The main conclusion is that these characteristics affect the creativity level. To confirm and compare this conclusion with others, we will use another test (Somers' D) to gain an additional perspective.

 **Table 17**: *Tb correlation coefficient values and significance levels between creativity and the project's features*

### **5.1.2 Somers’ Delta Test**

Somers’ D is a directional test (i.e., for two ordinal variables A and B, the A-B result are not the result of B-A) of association between two variables. Somers’ D results take values between -1 when all variables values disagree and 1 when all variables values agree. We defined creativity as the dependent variable and test it with every column in PrM. As a result, we found that multidisciplinarity, projects’ type, risk level, artistry, and research level scored as the highest Sd value with creativity as dependent variable .717, .248, .310, .631, and .403 respectively (see table 18).

**Table 18**: *Sd Somers’ delta values and significance levels between creativity and the project's features*

## **5.2 Model Evaluation**

According to the statistical tests, we selected the significant features (multidisciplinarity, projects' type, artistry, risk level, and research indicator) and evaluated an ordinal classification model (Frank & Hall, 2001). We used this version of classification because creativity, the dependent outcome, is an ordinal variable. The output is a probability vector, where the ith element is the probability the input belongs to class i. With these results, we can estimate a project's chances of being creative based on its characteristics.

Following is the classification process and its results. We created the classification process with randomly 65 projects and tested it with the remaining projects.

### **5.2.1 Ordinal Classification**

First, we note that due to the relatively low sample size (75 samples), the model is an initiative proposed model for estimating a project's level of creativity, and further research on an extensive data set is needed.

According to the suggested ordinal classification described above, we evaluated four binary classifiers. We randomly selected 65 projects out of the 75 projects and used their vectors as input to each classifier. The models transformed the problem from a five-class ordinal problem to four binary class problems. We tested the model with the remaining ten projects. The classifier estimated a relative creativity level in most cases.

There are cases where the classifier has no distinct choice between two levels of creativity, for example, an attempt to estimate a project with a creativity level of three. In some other cases, the classifier did not decide between the creativity levels but returned a probabilistic priority to the relevant creativity it tried to classify. There are cases where the classifier failed to estimate vectors, and as can be seen, these mistakes were made when the creativity level is one (see table 19).

**Table 19**: *Estimated creativity probability vector compared to the actual creativity level*

The four binary classifiers are decision trees of depth three, and each classifier contributes its decision to the probability output vector (see figure 18). We examined each classifier's influential characteristics with a scale of one to five, where one is the most noteworthy feature. It can be assumed that multidisciplinarity and artistry levels are the most homogeneous features in estimating creativity (see table 18). Generally, we can conclude that multidisciplinarity and artistry have a substantial effect on the creativity level (RQ4).

**Table 20**: *Features importance of the four binary classifiers. The first classifier (i.e., creativity level is one) rated artistry as the most noteworthy feature, while the other classifiers rated multidisciplinarity as the most noteworthy feature.*

**Figure 18**: *Visualization of the four binary decision tree classifiers*

# **6 Conclusions and Future Work**

This work presents an educational method to enhance creativity, multidisciplinarity, artistry, and collaboration among computer science students. The method is divided into three phases and uses a plugin-based platform as a creative education and collaboration tool. Throughout the development process, the students were introduced to the platform abilities and integral concepts such as separation of projects into independent plugins, handling and transferring data between plugins, collaboration, reuse of other developers' components, and more. We collected and analyzed the projects and the students' outcomes from the student and team perspectives.

The results show that multidisciplinarity, artistry, risk level, and project type are the projects' influential characteristics. Out of the five categories of "Computer Music" projects, Sonification was the riskiest project that combines multiple disciplines. Projects of that type were rated as most creative. Students who defined themselves as self-learners combined more disciplines in their projects than others. The plugins that were developed during the study were built based on components with a dedicated role and thus helped students understand software architecture more deeply. The computational analysis reinforces the claim that these characteristics influence creativity alongside whether it is a research project or not.

Future investigations are recommended to validate the kinds of conclusions that can be drawn from this study. Future studies could investigate creativity, multidisciplinarity, and artistry among students with the proposed method and strengthen or weaken the claims made in this work. The experiment proposed in this work can be repeated with many participants (during an academic course or a hackathon) to improve the ordinal classification model and accurate the estimated results. Also, in future studies, it is possible to test the collaboration process. With the platform's help, it can be done easily since the plugins are independent entities that also interface with the platform.

## **Ethics**

The studies involving human participants were reviewed and approved by IDC's Adelson School of Entrepreneurship Ethics Committee. The participants provided their written, informed consent to participate in this study.