TASMANIAN: Virtual lab software platform for teaching functional programming

Valentina MARASCU[1,2], Marius Iulian MIHAILESCU[1]

[1] Scientific Research Center in Mathematics and Computer Science,
Faculty of Engineering and Computer Science, SPIRU HARET University

[2] Low Temperature Plasma Department, National Institute for Laser,
Plasma and Radiation Physics

m.mihailescu.mi@spiruharet.ro

Abstract: *The need for efficient and interesting platforms that teach difficult programming paradigms is expanding quickly in the contemporary programming education environment. The full TASMANIAN platform described in this paper is created to make it easier to teach and master functional programming theoretical concepts and practical implementations. The platform, known as "TASMANIAN," uses interactive coding exercises, dynamic visualizations, and a user-friendly interface to overcome the difficulties of teaching functional programming.*

The main goal of the work is to present the TASMANIAN architecture, to underline its importance to students as an immersive learning experience that actively involves them into coding problems. The platform includes elements like a Coding Playground, Challenge Engine, Interactive Visualizations, and Modules focused on fundamental concepts in functional programming. Together, these elements give students the opportunity to practice pattern matching, experiment with functional data structures, and understand the subtleties of monadic composition.

Keywords: Functional Programming, Haskell, F#, F Sharp, Scala, Programming, Education, Virtual Lab.

1. Introduction

The field of programming education is constantly changing, necessitating creative methods for efficiently introducing complex programming concepts. Functional programming stands out among these paradigms for emphasizing immutability, higher-order functions, and compact code. Immersive and interactive platforms play a critical role as educators work to close the gap between theory and practice.

The TASMANIAN platform represents in our vision an important added value to programming education that is presented in this paper. To make functional programming principles easier to teach and learn, TASMANIAN was created. With

https://doi.org/10.58503/icvl-v18y202305

its dynamic visuals, interactive coding challenges, and user-friendly interface, TASMANIAN provides a stimulating environment that enables students to not only comprehend but also put functional programming principles into practice.

The rise of functional programming as a career field highlights the significance of strong educational resources. Traditional techniques frequently have trouble communicating both abstract ideas and real-world applications at once (Dobhal et al., 2023). By giving students a platform that combines theoretical instruction with direct hands-on experience, TASMANIAN overcomes this difficulty. TASMANIAN promises to provide students a thorough understanding and practical mastery of this paradigm by allowing them to experiment with functional programming ideas in real-time (Angione et al., 2023).

The importance of good instructional resources is brought home by the growth of functional programming as a career sector. Traditional methods sometimes struggle to convey both abstract concepts and their practical applications simultaneously (Zampetti et al., 2022). TASMANIAN gets around this problem by providing students with a platform that blends theoretical training with immediate hands-on practice. By enabling real-time experimentation with functional programming concepts, TASMANIAN promises to provide students a complete knowledge and practical mastery of this paradigm (Asakawa et al., 2022).

1.1 The future of functional programming

As technology landscapes change and programming paradigms advance, the future of functional programming promises intriguing possibilities. Future developments in functional programming may be influenced by the following major trends and directions:

- *Functional programming* ideas are likely to find their way into popular programming languages as they gain in popularity and understanding. Possible hybrid programming languages that support a wider range of programming requirements include both imperative and functional programming elements (Zampetti et al., 2022).
- *Parallel and concurrently programming* is possible thanks to functional programming's intrinsic emphasis on immutability and the absence of side effects. Functional programming may become the method of choice for dealing with concurrent issues as the demand for effective utilization of multi-core computers and distributed systems grows (Asakawa et al., 2022).
- Building *responsive and interactive applications* is becoming more popular thanks to functional reactive programming (FRP). To handle changing state over time, it makes use of functional programming principles. FRP may become a key strategy as user interfaces become more intricate and engaging (Borsatti et al., 2022).

- *Domain-Specific Languages (DSLs)*. The expressiveness and composability of functional programming make it suited for the development of DSLs. By offering specialized abstractions for problem domains, these languages can increase developer productivity and code readability (Bhat et al., 2022).
- Data-centric subjects like *machine learning and data science* are ideally suited to functional programming's emphasis on data transformations and composability. Functional programming libraries and tools may become more prevalent, especially those made specifically for analyzing and manipulating data (Zhang et al., 2022).
- *Cloud computing and serverless computing*. The serverless computing approach is consistent with the functional programming principles of immutability and statelessness. Building scalable, event-driven serverless applications might be a good fit for functional programming languages (Parham-Mocello et al, 2022).
- *Type Systems and Tooling*. It's expected that advanced type systems will continue to be incorporated, as seen in languages like Haskell. Enhanced type systems can detect problems at build time, resulting in codebases that are more dependable and maintainable. Functional languages may also become easier for developers to use with better tooling and IDE support (Izuta et al., 2021).
- *Education and Training*. Education and training resources will increase as the need for functional programming abilities rises. A greater pool of qualified functional programmers will be produced because of online courses, tutorials, and interactive platforms (Maguerra et al., 2021).

Functional programming has a bright future ahead of it in many areas, from general use to niche applications. Because of its focus on modularity, immutability, and composability, it is well-suited for addressing the issues of contemporary software development, making it a useful and influential programming paradigm for years to come (Asakawa et al., 2022).

1.2 Paper contributions

The paper brings important contributions to the field of how functional programming can be taught at classes by introducing the TASMANIAN Virtual Lab Software Platform. These contributions can be summarized as follows:

- *Innovative Learning Environment*. The paper introduces a novel learning environment, TASMANIAN, designed to teach functional programming concepts. This platform offers an interactive, immersive, and hands-on approach to learning, enabling students to actively engage with

functional programming principles.

- *Effective Pedagogical Tool***.** TASMANIAN addresses the challenge of teaching abstract programming paradigms by providing interactive coding challenges, dynamic visualizations, and instant feedback mechanisms. These features enhance comprehension and retention of functional programming concepts.
- *Bridge between Theory and Practice***.** The platform's combination of theoretical explanations and practical exercises bridges the gap between theoretical understanding and real-world application. Students can not only grasp the theory but also immediately apply it in a controlled coding environment.
- *Comprehensive Content*. TASMANIAN covers a range of functional programming topics, including functional data structures, pattern matching, monadic composition, and more. Its modular design allows learners to progress from fundamental concepts to advanced topics in a structured manner.
- *Immediate Feedback and Analytics***.** TASMANIAN's Feedback and Analytics module provides students with instant feedback on code correctness, promoting an iterative learning process. The platform's analytics capabilities offer insights into performance metrics, enabling learners to track their progress over time.
- *Promising Tool for Educators*. For educators, TASMANIAN presents a powerful tool to facilitate the teaching of functional programming concepts.
- *Real-World Relevance*. By providing an environment for experimentation and application, TASMANIAN helps students understand the practical implications of functional programming.

TASMANIAN can dramatically improve the learning process and mastery of functional programming principles by addressing the difficulties of teaching abstract topics, encouraging engagement, and offering right away application opportunities.

2. Background and motivation

With an emphasis on immutability, higher-order functions, and declarative coding, functional programming has become a potent paradigm. Although its advantages are generally known, teaching functional programming ideas successfully is still difficult. Traditional methods frequently have trouble bridging the gap between theoretical knowledge and real-world application.

Innovative strategies that enable students to not only conceptually

understand these concepts but also experience their implementation in real-world circumstances are needed to address these issues. Instructors are looking for teaching resources that offer dynamic interactivity, immediate feedback, and engagement incentives so that students may develop a solid foundation in functional programming.

The *goal of this work* is to present a remedy that fills the gap between theoretical understanding and hands-on training in functional programming education. By developing an engaging and interactive learning environment, the proposed TASMANIAN platform aims to improve the way functional programming is taught. We shall examine TASMANIAN architecture, attributes, advantages, and prospective influence on the area of programming instruction in the parts that follow. The goal of TASMANIAN is to make functional programming approachable, engaging, and transformative for students of all levels by addressing the drawbacks of conventional teaching techniques and utilizing the benefits of interactive learning.

3. TASMANIAN: Virtual Lab Software Platform

TASMANIAN is a cutting-edge Virtual Lab Software Platform created to transform the way functional programming principles are taught and learned. Students and developers alike may actively engage with the concepts and techniques of functional programming thanks to its dynamic and immersive environment.

The platform (see Figure 1) is composed from 14 modules.

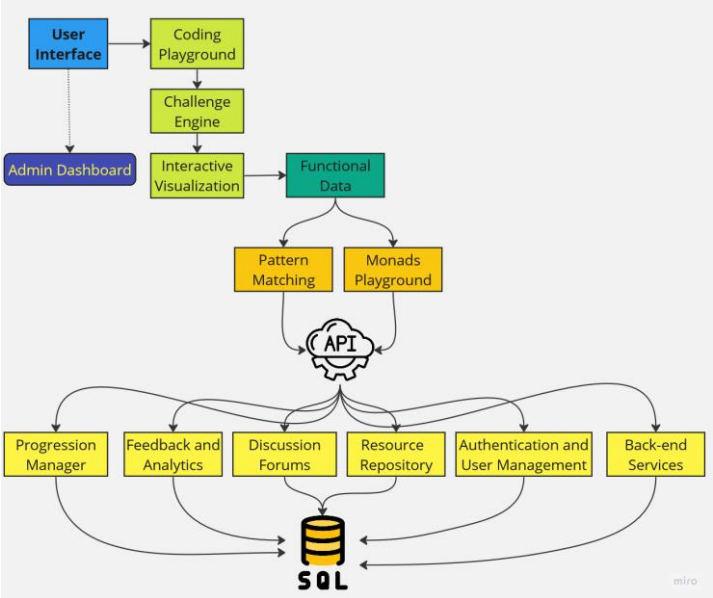

Figure 1. TASMANIAN Platform

These modules are described as follows:

- *Coding playground*. The novel approach of TASMANIAN platform is created to transform the way functional programming principles are taught and learned. It functions as an interactive and immersive environment that enables students, whether they are developers or students, to engage actively with the ideas and techniques of functional programming.
- *Challenge engine*. The module provides interactive coding problems, is essential to TASMANIAN's effectiveness. The challenges include a wide spectrum of functional programming principles and complexity levels, allowing students to develop their skills over time.
- *Interactive Visualizations*. To show abstract ideas, data changes, and code execution, TASMANIAN uses dynamic visuals. Visualizations give consumers a visual depiction of how functional programming functions, which aids in their understanding of complicated concepts.
- *Functional Data Structures Module.* Students of this module can construct and modify useful data structures including lists, trees, and maps. With these structures, students can test how immutability affects data management.
- *Pattern Matching Simulation.* TASMANIAN provides a simulator that enables students to practice pattern matching on different data types to make learning pattern matching easier. It aids students in understanding the complexities of disassembling and comparing data structures.
- *Monads Playground.* The module provides a controlled environment for students to experiment with monads and understand their role in managing side effects. Students can explore monadic transformations and compositions.
- *Progression Manager*. The module is responsible for tracking the student progress and recording the achievements obtained by student. Also, it has the possibility to unlock new challenges based on completion.
- *Feedback and Analytics*. The student after he perform his exercises and uploads its source code, the module will generate feedback based on the code correctness. Within this module we will have the possibility to collect student data for performance analysis.
- *Discussion Forums*. This is an essential space for students where they can ask questions, share solutions, and interact with other students and instructors.
- *Resource Repository*. Responsible for storing additional learning

materials, tutorials, and bibliography/references sources links. The repository will help in enhancing the student learning experience.

- *Authentication and User Management*. The module is dedicated for handling student registration, login, and profiles. The individual progress will be tracked and personalized content is offered to the student.
- *Database*. Within database will store user profiles, progress, and different amount of data specific to student achievements. The database will help in supporting tracking and analytics.

4. Key features and benefits

TASMANIAN includes a variety of interactive tools to improve learning and increase understanding of functional programming ideas.

Personalized learning paths and rapid feedback are two significant interactive features: *Instant Feedback*. TASMANIAN gives immediate feedback regarding the effectiveness and correctness of user-written code. In the platform's coding environment, TASMANIAN displays syntax issues, logical errors, and areas for optimization as students write and run code. Students may immediately recognize and remedy mistakes because to the instant feedback method, which encourages self-correction and experimentation-based learning. *Personalized Learning Paths*. TASMANIAN uses adaptive algorithms to evaluate the proficiency and learning progress of students. The platform customizes the learning experience for each user based on initial assessments and performance on coding challenges. Students are neither too nor underchallenged since challenges and content are offered that match their level of skill. TASMANIAN dynamically modifies the difficulty of challenges as students improve and show mastery, enabling ongoing skill development.

Dynamic visualizations are extremely important for improving understanding of abstract ideas, especially while learning functional programming. *How do they accomplish this?* The answer can be summarized through the followings: *Concrete representation of abstract ideas* - functional programming principles that would otherwise be abstract and difficult to understand are given a concrete, visual representation through dynamic visualizations. They can, for instance, demonstrate how data transformations take place within a functional pipeline, which will help students better understand how functions manipulate data step by step. *Immediate feedback* - visualizations provide real-time feedback by displaying how data or program state changes while code is executed. Students may immediately observe the results of their code, strengthening the relationship between the code and its effects. *Intuitive understanding* - by providing functional programming principles in an interactive, graphical way, visualizations assist students in developing an intuitive understanding of them. Students may "see" how

functions relate to data, how immutability is upheld, and how pattern matching breaks down intricate structures. *Conceptual mapping* - visualizations create a visual map of relationships between various components of functional programming. They illustrate how functions can be composed, how data flows through pipelines, or how recursion operates, helping learners connect the dots between abstract ideas. *Experimentation and exploration* - Experimentation is encouraged by dynamic visualizations. Students can update the code and see how the changes affect the data and outcomes. Students can test theories and investigate variants because of this experimentation, which promotes active learning and curiosity. *Error Identification* - error identification is facilitated by visualizations. The visual depiction allows learners to identify abnormalities or unexpected behaviors, which helps with debugging and problem-solving. *Contextual learning* - visualizations provide context to code snippets. Learners can see the bigger picture of how functions fit into the functional programming paradigm. This contextual understanding is particularly valuable when transitioning from imperative to functional programming. *Engagement and retention* - visualizations are inherently engaging. They break the monotony of text-based learning and can increase retention of complex ideas. Learners often find visual content more memorable and motivating.

5. Case study

We will consider the following case study: Teaching functional programming ideas to a wide collection of students, ranging from novices to experienced programmers, was a regular challenge for the computer science department at a university. They are looking up for a solution that could reconcile theory and practice while considering a range of skill sets.

To teach functional programming, the department chose to incorporate, into their curriculum, the TASMANIAN as a virtual lab software platform. For students enrolled in functional programming classes, TASMANIAN is available as an optional resource which will have the following benefits:

- *Accessibility for all skill levels*. The customized learning routes offered by TASMANIAN should be quite helpful. Beginner students will be given foundational exercises to complete, while more seasoned students were given access to advanced courses. All students were able to use the platform at their own speed and skill level because to its versatility.
- *Interactive learning and immediate feedback*. The coding environment and the fast feedback features were quite popular with the students. They might play with the ideas of functional programming while getting prompt direction. Students were inspired to actively contribute and refine their code because of the interaction, which led to a deeper comprehension of the ideas.

- *Visualizing abstract concepts*. Dynamic graphics created by TASMANIAN were helpful in making abstract concepts clear. Students can see how data moved via functional pipelines, how functions changed data, and the advantages of immutability. Students were able to relate theory to practical situations with the use of visualizations, which made functional programming more understandable.
- *Collaborative learning*. Peer-to-peer learning was made possible via the discussion boards in TASMANIAN. Students are having the possibility to post queries, exchange knowledge, and work together to solve difficult challenges.
- *Improved learning outcomes.* In functional programming courses, students who are regularly using TASMANIAN showed enhanced learning outcomes. Their comprehension of fundamental ideas like monads, pattern matching, and recursion was much improved. To achieve these successful results, the platform's real-time feedback and tailored learning paths were key factors.
- *Instructor insights.* Instructors will be able to observe that students who participated in TASMANIAN showed greater confidence in their ability to apply functional programming ideas to projects and tasks for their courses.

The department's problem of teaching functional programming to a diverse student group will be successfully met through TASMANIAN. The learning process will be improved by its adaptability, interaction, visualizations, and community involvement. The platform will help students to better understand complex ideas while also boosting their self-confidence and performance in functional programming classes.

6. Future research directions

As future research directions, we will focus on including other modules such as the ones described below, focusing on developing applications for IoT, Edge Computing, Functional Programming for Quantum Computing, formal verification, and program analysis and much more.

The following future research directions for TASMANIAN platform are considered genuine and with a considered added value for such platform. These are: advanced functional programming paradigms, formal verification and program analysis, concurrency and parallelism in functional programming, human-computer interaction (HCI), functional programming for IoT and Edge Computin, functional programming for machine learning, functional programming for quantum computing, and functional programming for education. A partnership with industry will be required as well for providing real-word applications.

7. Conclusion

The paper introduces TASMANIAN a Virtual Lab Software Platform meant to transform functional programming idea teaching and learning. The following is a summary of the paper's main ideas and contributions:

It is impossible to overstate the importance of TASMANIAN in the field of functional programming education. It offers a revolutionary answer to the persistent problems with teaching abstract ideas. Functional programming wish to be more approachable and interesting for learners of all backgrounds because to the platform's interactive features, tailored learning routes, instant feedback systems, and visualizations.

TASMANIAN gives instructors the tools they need to close the gap between theory and application so that students may not only comprehend but also successfully apply the concepts of functional programming. Its capacity to influence the future of programming education is highlighted by its effectiveness in enhancing learning outcomes, which has been proven by case studies.

REFERENCES

Angione, F., Bernardi, P., Di Gruttola Giardino, N., Appello, D., Bertani, C. & Tancorre, V. (2023). A guided debugger-based fault injection methodology for assessing functional test programs. In *IEEE 41st VLSI Test Symposium (VTS), San Diego, CA, USA*. pp. 1-7, doi: 10.1109/VTS56346.2023.10140099.

Asakawa, K. & Tanaka, T. (2022): Visual Programming environment for learning functional programming using unit test. In *12th International Congress on Advanced Applied Informatics (IIAI-AAI), Kanazawa, Japan*. pp. 220-223 doi: 10.1109/IIAIAAI55812.2022.00051.

Bhat, S. & Grosser, T. (2022). Lambda the Ultimate SSA: Optimizing Functional Programs in SSA. In *IEEE/ACM International Symposium on Code Generation and Optimization (CGO), Seoul, Republic of Korea.* pp. 1-11, doi: 10.1109/CGO53902.2022.9741279.

Borsatti, D., Cerroni, W. & Clayman, S. (2022) From Category Theory to Functional Programming: A Formal Representation of Intent. In *IEEE 8th International Conference on Network Softwarization (NetSoft), Milan, Italy*. pp. 31-36, doi: 10.1109/NetSoft54395.2022.9844061.

Dobhal, D. C., Kumar, B. & Das, P. (2023). Involvement of Functional Programming in Language Processing and Machine Learning. In *2nd International*

Conference for Innovation in Technology (INOCON), Bangalore, India. pp. 1-4, doi: 10.1109/INOCON57975.2023.10101253.

Izuta, R., Matsumoto, S., Igaki, H., Saiki, S., Fukuyasu, N. & Kusumoto, S. (2021) Detecting Functional Differences using Automatic Test Generation for Automated Assessment in Programming Education. In *28th Asia-Pacific Software Engineering Conference (APSEC), Taipei, Taiwan*. pp. 526-530, doi: 10.1109/APSEC53868.2021.00062.

Maguerra, S., Boulmakoul, A. & Badir, H. (2021) Time Framework: A Type Level and Algebra Driven Design Approach. *IEEE Xplore*. doi: 10.1109/ICDABI53623.2021.9655926.

Parham-Mocello, J., Erwig, M., Niess, M., Nelson, A., Weber, J. & Berliner, G. (2022) Using a Functional Board Game Language to Teach Middle School Programming. In *IEEE Frontiers in Education Conference (FIE), Uppsala, Sweden*. pp. 1-9. doi: 10.1109/FIE56618.2022.9962569.

Zampetti, F., Belias, F., Zid, C., Antoniol, G. & Penta, M. D. (2022) An Empirical Study on the Fault-Inducing Effect of Functional Constructs in Python. In *IEEE International Conference on Software Maintenance and Evolution (ICSME), Limassol, Cyprus*. pp. 47-58, doi: 10.1109/ICSME55016.2022.00013.

Zhang, Z.-D., Gao, X., Luo, J., Zhong, Y.-N., Xu, J.-L. & Wang, S.-D. (2022) UV-Enabled Multibit Organic Transistor Memory with High Controllability and Stability. In *IEEE Electron Device Letters*. 43(1), 124–127. doi: 10.1109/LED.2021.3132996