An Adaptive System Architecture Model for the Study of Logic and Programming with Learning Paths

Adson M. da S. Esteves[1], Aluizio Haendchen Filho[2,3], André L. A. Raabe[1] and Rudimar L. S. Dazzi[2]
[1]*Laboratory of Technologic Innovation in Education, University of the Itajaí Valley (UNIVALI), Rua Uruguay, 458, Itajaí, Brazil*
[2]*Laboratory of Applied Intelligence, University of the Itajaí Valley (UNIVALI), Rua Uruguay, 458, Itajaí, Brazil*
[3]*Univertity Center of Brusque (UNIFEBE), Brusque, Brazil*

Keywords: e-Learning, Intelligent Tutoring Systems, Adaptive System, Constructivism, Constructionism.

Abstract: The number of dropouts and evasion rates in computing courses are among the highest in Brazilian universities. To reduce this rate, eLearning technologies are being used to compose solutions. Because of such reality, this work aims at showing an adaptive system architecture with learning paths that best fit the student's profiles and interests. In order to take student's profiles and interests into account, two theories will be used: constructivist and constructionist. The fundamentals of these theories were analysed to formulate a teaching structural model for the system. The literature was researched to find adaptive systems with theories similar to constructivism and constructionism. Then it was designed a collaborative agent system based on intelligent software agent techniques to help the student on its paths and content choices. In this system, the student's difficulties, characteristics, and knowledge obtained from other users can be reused. An environment with a content hierarchy which allows more attractive learning path construction options may ease the learning, make the study more interesting and help reduce evasion rates in computing courses.

1 INTRODUCTION

Dropout is a negative phenomenon present in Brazilian higher education. Related to the negative phenomenon, it is possible to explicit: the students themselves, the teaching institutions, and the market. For students who drop out of college, it reduces their chances of personal and professional growth. For the institutions, they fail to fulfil their social function of educating, on the one hand, and fostering the labour market, on the other. Related to the market, it is noteworthy that it is increasingly interested in new professionals related to the areas of computing and information technology (IT), as shown by various job search sites (Guia da Carreira, 2018; MichaelPage, 2019; Pattabiraman, 2019; CareerCast, 2019; Trade Schools 2019). Young people seeking professional growth in this market choose computer-related courses to meet this demand.

Even though there is a good number of students entering these courses, the dropouts in Brazil are quite high. Dropout rates can range from 22% to 32% and is the second highest rate among courses at Brazilian universities (Filho et al, 2007; Lobo, 2017). Among the reasons that lead to this, it is the difficulty of the initial subjects that are the pillars of the course: Algorithms and Programming. These subjects are considered difficult by many beginners because of the need for the required abstraction and logical-mathematical skills they do not have in their daily life (Raabe and da Silva, 2005).

Seeking to better understand this problem, Giraffa and Mora (2016) conducted a survey with students who stopped attending computer science courses at PUCRS during the period of 2012-2013. In this research, it was found that the main reasons that contributed for the dropouts were: (i) lack of study time, as many have working hours; (ii) difficulties in understanding issues in the classroom and in activities; and (iii) teachers not so well qualified to serve several students. Such difficulties point not only to the problem of lack of prior skills, but to the didactic organization of teachers and the formulation of activities, as well as students' individual problems.

Teachers specific to each individual student would be ideal for solving problems, but the cost makes this practice unfeasible (Weragama, 2013). In addition, the difficulty of classroom issues may be

Esteves, A., Filho, A., Raabe, A. and Dazzi, R.
An Adaptive System Architecture Model for the Study of Logic and Programming with Learning Paths.
DOI: 10.5220/0009412406790690
In *Proceedings of the 22nd International Conference on Enterprise Information Systems (ICEIS 2020) - Volume 1*, pages 679-690
ISBN: 978-989-758-423-7

related to the different types of students entering computer courses. Heterogeneous profiles such as gender, age, education level, way of learning, and problem-solving skills make it difficult to create unique content that addresses them all (Oliveira et al, 2015).

In order to reduce the problem, two points are highlighted: changing teaching methodologies and E-learning.

E-learning is a web-based learning ecosystem integrating several stakeholders with technology and processes. It provides a flexible and personalized way to learn, allowing learning on demand and reducing its cost (Cidral, 2018).

The application of E-learning is possible with the Intelligent Tutoring Systems (ITS). Nowadays, also called Adaptive Systems, they incorporate Artificial Intelligence (AI) techniques to develop tutors who know what, for whom and how they teach (Nwana, 1990), that is, they consider the peculiarities of the student.

On ITS development, many authors have successfully used Multi-Agent Systems (MAS) proposed (Giraffa, 1999; Yaghmaie and Bahreininejad, 2011; Dolenc and Aberšek, 2015; Hooshyar *et al*, 2015; Harley *et al,* 2015; Vaidya and Sajja, 2016). Using MAS can help with complex tasks such as monitoring student activity, capturing information about their dynamic contexts, recommending content based on their profiles, and more (Frade, 2015).

Another way to solve problems may be by changing teaching methodologies. Some suggest a change in the form of general education to facilitate learning, such as constructivism (Piaget, 1967) and constructionism (Papert, 1980). In the constructivist approach, during the learning process, the student may be able to decide how to learn and act proactively in knowledge building (Bada 2015). In addition, constructionism points out that building a product related to students' interests helps learning to occur in a more efficient way.

One way to enable students to conduct their learning in computing is to provide choices on learning paths. Paths are different types of skills or knowledge that follow a user-definable sequence. Thus, led by the adaptive system, the students can define their own learning path, facilitating the acquisition of the content.

Learning environments can be tools for applying constructionism (Baranauskas, 1999). They can be applied with microworld building, hypermedia text, and programming environments. Programming environments that help the user and facilitate the learning of logic present constructivist characteristics.

This paper presents an adaptive system architecture model with learning paths for programming teaching. For the development of learning paths, the IDE Portugol Studio (Noschang, 2014) is used. It was developed with the purpose of facilitating the learning of programming logic in Brazil and has architectural features that facilitates to create adaptable learning paths for the students.

The adaptive system is designed to provide the following features: (i) learning paths with adaptive options, led by an intelligent tutor so that students can choose the knowledge of interest; (ii) adapted resolution tips and support materials during the exercise, if the student has difficulties; and (iii) providing exercises adapted to the student's personal interests. The proposed solution utilizes MAS technology, and the MIDAS platform provides infrastructure services such as communication, management, and interaction between agents.

2 BACKGROUND

The following subsections describe the main foundations used for solution development.

2.1 e-Learning and Adaptive Systems

Considering the rapid growth of Technology and Population, it seems inevitable that E-learning is going to be the main agent for education. It involves innovative pedagogy, advanced teaching strategies and learning methodologies and approaches tightly connected with flexibility, accessibility, openness, and communication through modern services. Personalized intelligent agents and recommender systems have been widely accepted as solutions towards overcoming information retrieval challenges by learners arising from information overload (Tarus, 2017).

That's why Intelligent Tutoring Systems are great systems to integrate into E-learning environments (Phobun, 2010). For years, there have been several proposals on how ITS can be modelled. The following works have similar themes and ideas that helped in the development of the architecture proposed in this work.

Cabada *et al* (2017) developed Java Sensei, an adaptive learning environment based on student preferences. The system recommends exercises based on other students' grades who have previously used the system. It uses sentiment analysis captured through cameras to provide personal instruction to users. It also allows the creation of exercises without the need to program. The system architecture consists of the following layers: (i) presentation, which is the interface that communicates with the user; (ii) tutor, comprising a domain module, a tutor module, a student module and an exercise recommendation system; (iii) intelligence, which infers the characteristics of the students using the system by analysing their emotions; (iv) web content, which contains key files for the logic of previous layers; and (v) data, which contains the student and domain saved data.

De Meo et al (2007) used the MAS technique to develop the X-Learn. The virtual learning system is structured in XML. The system consists of 3 main agents: (i) User Device Agent (UDA); (ii) Skills Management Agent (SMA); and (iii) Learning Program Agent (LPA). Knowledge is stored in a learning object repository. By activating the UDA associated with it, the student has access to a list of skills they can learn. This list is made in collaboration with the SMA, which analyses the skills he has already learned, and the skills he can learn. After the user selects what they want to learn, SMA brings up a list of subjects needed to learn that specific skill. It then cooperates with LPA to define the best learning program based on knowledge and user profile information.

Panagiotis *et al* (2016) developed the APLe (Agents for Personalized Learning). They based their system in 2 ontologies: (i) Learner Model and (ii) Learning Object and Outcomes. The first models the student's characteristics like learning and social style, use of technology, literacy, experience, time of study and reasons for education. The second stands for the knowledge domain. It used Learning Objects, which are units of educational digital content and Learning Outcomes which are what the student is expected to know. The tutor Agents designated to each student uses a LSM (Learning Space Management) and a LTC (Learning Tactic Control). The LSM is the knowledge to be learned; it used learning objects to create a graph structure. The Learning Objects and Outcomes are placed in this graph with the actual states of the learner. They tell if the student learned something depending on the states they are. The LTC is a reactive component that selects the tactic to apply to the LSM. When the learner asks for a recommendation, the system triggers the LTC to use a formula to balance the LSM nodes and determine which nodes of content should be recommended.

All these works use Multi-Agent Systems. While they are not obligatory to make them function, MAS allows for more scalability, error tolerance, robustness and security. These works help formulating some structures from the proposed system as it will be seen in sections 3 and 4.

2.2 Constructivism and Constructionism in Learning Paths

The constructivism, proposed by Piaget (1980), is based on the principle that learning is a process of knowledge construction in which the student is an active part of the process. Children not only record what they are taught, but also formulate their own ideas with prior knowledge. Previously acquired knowledge is not necessarily wrong, but with experience, by validating this knowledge, it is possible to correct and define new solutions.

By giving the learner a leading role in learning, during the process of knowledge building constructivism allows one to confront the results of experiments and re-evaluate their ideas about how the world works (Philips, 1995).

However, Papert (2008) studies Piaget's research and criticizes some aspects. He points out how concrete learning, described by Piaget, is a subjective term that leads to misinterpretation. Papert proposes constructionism as a form of "reconstructed" learning of Piaget's constructivism.

For Papert, learning is not just about experimentation, building objects in the world plays a more important role in knowledge building. For him, the construction is not only theoretical, but also practical of a real product. Moreover, the context of this construction being relevant to the student allows the construction of the mental model itself to be facilitated by association with a subject that already knows.

To provide the student with the possibility of construction and protagonism, the chosen form was the use of learning paths. These paths are like a sequence of nodes, where each node is related to a skill or subject learned. These subjects and skills have various contents within them that serve as a learning possibility for the student. The student who wants to learn some skill can look at the contents and look for the one of their choice. A student looking at various content in sequence will build a content path, ultimately called the learning path. Details of the path structure can be seen in section 4.1.

Raabe *et al* (2016) points out how contexts applied in exercise statements can influence the resolution of activities, as they show a real application of this knowledge. This, however, cannot be generalized since students have their own context. If an exercise that is not in a context is applied, that application may not be relevant to learning. Thus, in addition to the paths, each tree node contains a series of content with different contexts to learn. The tutor suggests to the student the one with the best context for him to learn, depending on his interests.

2.3 Portugol Studio

Many authors propose tools that facilitate and remove barriers in early learning. There are several environments in the literature that help beginners in programming (Cooper, 2000; Ng, 2005; Resnick, 2009; Wolber, 2011; Paiva, 2016; Romagosa 2019) and Portugol Studio (PS) (Noschang 2014) is an IDE focused on Brazilian learners.

PS is a beginner-oriented programming IDE. The language syntax is defined in Portuguese and has a simplified interface for easy learning. Even as a beginner IDE, it has several libraries that allow more complex programs to be developed.

The IDE has a language like C and PHP. It has syntax documentation tabs, and library documentation in the help tab. It has several examples of programs already made, from simpler programs with "Hello World" writing to complete games using the existing graphics mode. It also has didactic error messages as standard which can help smart tutors. These messages tell the user the location of the error, why the error occurred, and examples of troubleshooting.

Some works that enable more effective intelligent tutoring have already been developed for this tool (Pelz, 2011; Hodecker, 2014). They deal with the development of an automatic exercise corrector as a plugin for PS.

The PS platform is open source and is already used in more than 7 universities. It has over 200,000 downloads and keeps up to date. Besides being able to change IDE source code, it is also possible to develop plugins, create buttons with functions in the code editor, add libraries, among others.

3 PROPOSED SOLUTION

The system was designed with features of an adaptive STI. It offers content on various topics with different themes. This enables the student to choose the content they want to learn and set up their own learning path. Meanwhile the system helps the student to choose the content that fits his profile. In addition, the system provides solving tips and support materials when performing exercises tailored to their profile if the students experience difficulties. The following subsections present the learning path structure, the generic architecture, and the proposed role model for the agents.

3.1 Path Model

The path model developed for the system uses a level and topic structure. In the system domain there are several different topics of programming learning, however some need previously taught topics to be learned. Because of that, the framework uses levels to define the order in which some topics need to be learned. This structure can be seen in Figure 1.

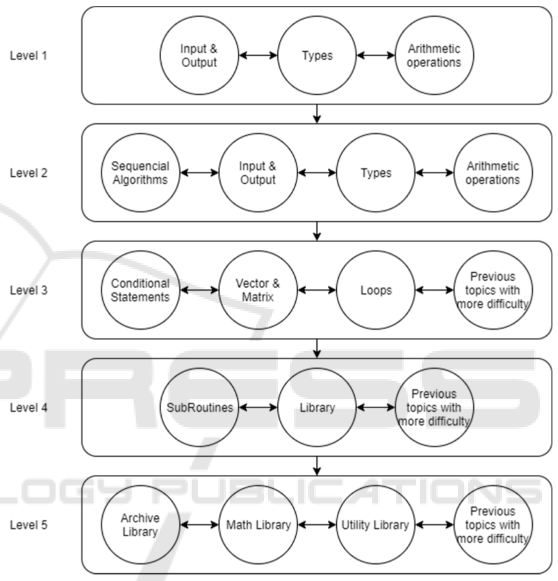


Figure 1: Structure of knowledge domain levels.

Existing topics at the same level can be learned in any order. Conversely, topics that are at different levels need to be learned according to the order of the levels at which they are located, starting from the lowest level to the highest level. The student using the system will need to complete all topics at the same level to advance to the next one. However, this does not mean that he will need to complete all existing content on that topic.

This is possible because each topic has several contents that validate the topic as complete if used by the student. These contents may vary between: (i) exercises and (ii) support materials. The contents have attributes that represent their characteristics that differentiate them from each other. Attributes range from: (i) Difficulty, (ii) Complexity, (iii) Content Type, (vi) Taxonomy, and (v) Tags. This can be seen in Figure 2.

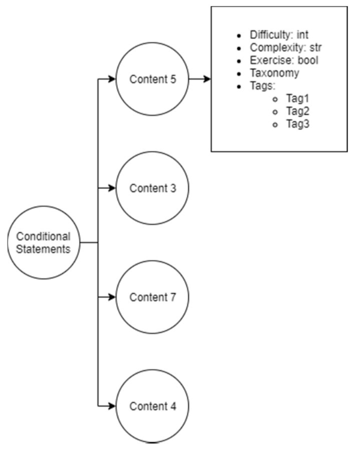


Figure 2: Topic structure, and knowledge domain exercise.

The tags in each content are related to the themes that the content addresses outside the programming. These themes are related to different preferences that students may have when using the system. For example, a student may prefer songs, so music-tagged content will be recommended to them.

When selecting content, students will have their option recorded in their account and linked to their previous content. This connection in sequence of contents is called the learning path. This can be seen in Figure 3.

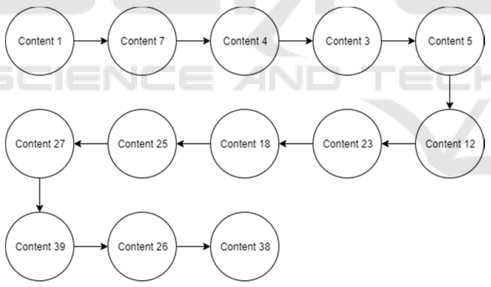


Figure 3: Generic example of a learning path.

The topics in a path do not necessarily have to be in order of levels. Topics that have already been completed by the student can always be revisited, so lower level exercises after higher levels can occur on the paths. This can happen in cases where the student realizes or is driven by the system that needs to reinforce content they should have already learned. The students, in these cases, will be directed to a different exercise of the same topic and placed on their path.

The track is recorded in the system, allowing it to be reused to help new students select their content. Similarities between student characteristics may identify paths that best suit them because they have already been used and successful by previous students.

3.2 Generic Architecture

The generic architecture is an abstract representation of the system, as shown in Figure 2. In Localhost, it is the PS system with its functions (AutoCorrect and Error Messages). The proposed solution is represented in two modules: the Web Platform and the Tutor Plugin.

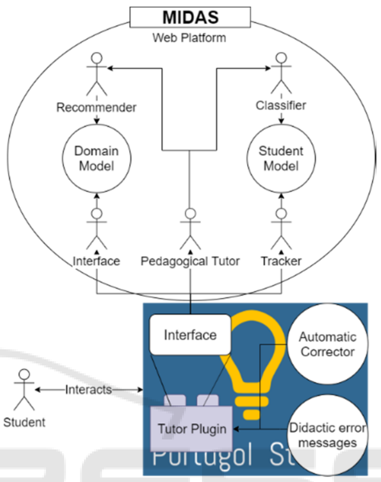


Figure 4: Proposed system generic architecture.

Domain and student models are contained in the web platform. They keep the content and student data respectively. Structured content data for learning paths as well as students' doubts registered in the system are stored in the domain model. In the student model, the consolidated profile data of all students in the system is stored.

Students using the system must create an account so that their profile and usage information is saved and help the tutor develop their teaching method. For the adaptive system to learn over time, information from all students using the system must be consolidated into a database.

In the web platform, the following five agents act:

(i) Pedagogical Tutor, who collaborates with Classifier and Recommender agents to provide students with exercises recommended by their preferences and similarities with other students;

(ii) Interface, which interacts with the student by recording their questions and answering them, and by creating an interface for the system administrator and the student;

(iii) Tracker, which is responsible for capturing student interactions with the system and storing them in the database;

(vi) Classifier, which is responsible for clustering students according to their interactions and preferences;

(v) Recommender, to recommend to the tutor module the best student exercise based on other students. With these agents working together, we seek to meet the needs of students in programming learning.

The web platform uses the MIDAS platform (Haendchen Filho, 2017) responsible for the execution of the multi-agent system, the management of communication between agents and the mechanisms of access to the database. Information on the operation of this system is described in section 2.1.

3.3 Tutor Module Specifications

The specifications of the agents that make up the Tutor Module were defined using the role model. This model is used in several approaches (Gonçalves 2009; Haendchen Filho, 2019) to provide a summary of agent activities. According to theory, a role can be described by two basic attributes: (i) Responsibilities: the role of obligations that indicate functionality; and (ii) Permissions: the rights associated with the role, indicating the features that the agent can use. Table 1 presents the role model of the agents.

Table 1: Agent Responsibility Table.

Organization: Adaptive System		
RESPONSABILITIES	PERMISSIONS	COLABORATION
Pedagogical Tutor Agent		
Suggest next contents to the student	Domain Model	Classifier
Suggest exercise resolution tips	Student Model	Recommender
Identify exercises errors		Tracker
Identify preferences		
Identify current knowledge		
Calculate Average Exercise Resolution Time		
Sort resolution time by subject type		
Classifier Agent		
Cluster students by their characteristics	Student Model	
Recommender Agent		
Recommend content by a group	Domain Model	
Tracker Agent		
Store clicks on interface	Student Model	
Store log time		
Store exercise solving time		
Store solved exercises		
Store chosen tracks		
Interface Agent		
Create Interaction Interface to the Admin	Domain Model	
Store user questions	Student Model	
Answer user questions		

The Tracking Agent is responsible for capturing student interactions with Portugol Studio. It is located on the web platform and receives user interaction data through the plugin interface. This agent is therefore responsible for recording in the student module the following student information: (i) clicks on the interface; (ii) length of stay in the system; (iii) exercise response time; (iv) solved exercises; (v) chosen paths.

The Interface Agent is responsible for formatting the interface for the system administrator. It has access to domain and student module information to allow the administrator to see this information on their screen. It also receives questions from users through an existing plugin interface. When a user asks a question, the agent searches the domain module for answers to similar questions and show the student. If they do not have similar questions already logged in the system, it will be logged until an administrator answers it. Therefore, their responsibilities are: (i) to record student's questions; (ii) notify the administrator about unresolved student's questions; (iii) send the students who asked the questions the respective answers of the administrators; (iv) answer student's questions without the need for an administrator if a similar question has already been asked by another student and answered.

The Tutor Agent is the one who chooses the best teaching strategy for the student. It has access to student module and domain module information. This agent collaborates with the Classifier agents to cluster students and discover similarities, and the Advisory agent to suggest student-tailored content. This enables new students to learn from the tutor's experiences with previous students. In addition, to better identify these groups, it identifies errors in exercises by enumerating by types of errors, and students' personal preferences and content selection. Finally, it analyses the reading time and resolution of each content, identifying which students spent more and less time. Thus, this agent's responsibilities are: (i) to suggest learning paths based on student preferences and profile (collaboration with Classifier and Recommender Agent); (ii) suggest real-time exercise solving tips with the aid of the automatic correction module to help you learn programming content more easily; (iv) identify errors in exercises (collaboration with Tracking Agent); (v) identify student preferences according to chosen exercises (collaboration with Tracking Agent); (vi) identify the average exercise resolution time (collaboration with Tracking Agent).

The Classifier Agent uses Clustering techniques to group students with similar characteristics, and Machine Learning techniques to identify students with potential dropout characteristics. The Classifier

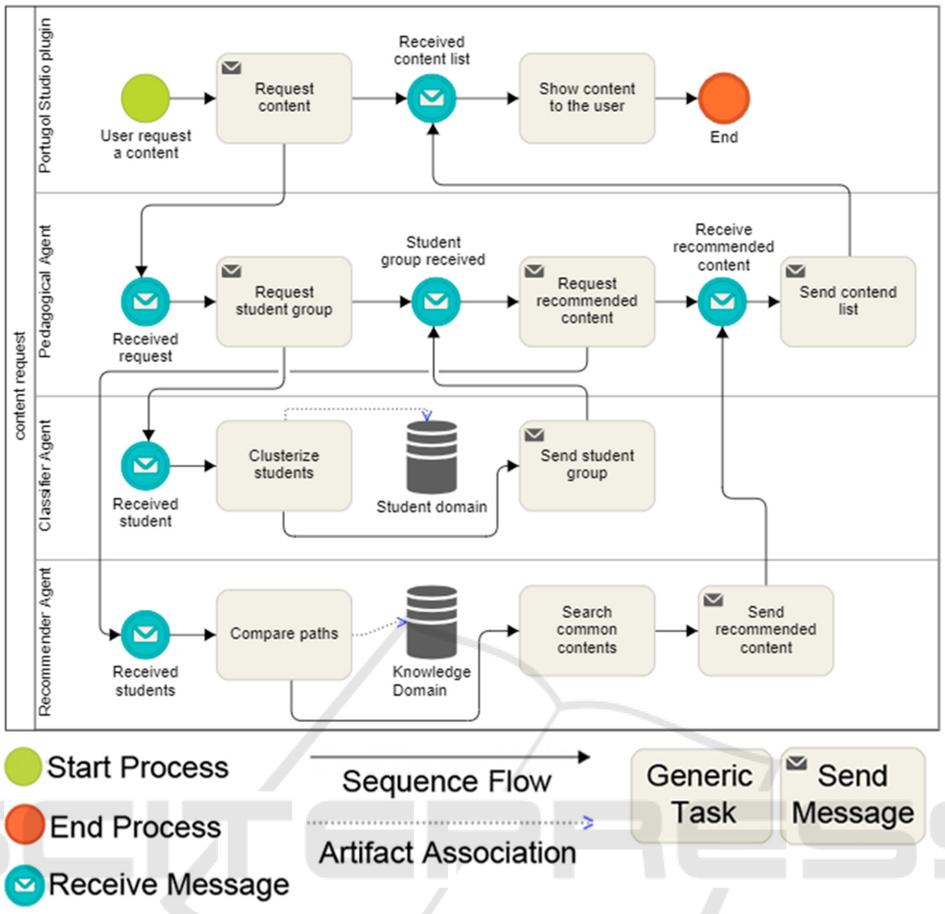


Figure 5: Content request process in BPMN model.

collaborates with the Pedagogical Agent. Once the student is classified into a group, you can check which exercises best fit them and their preferences. The profile data that allow classification are: (i) age, (ii) gender, (iii) educational level, (iv) personal preferences, (v) chosen paths, (vi) resolved exercises, (vii) average time exercise resolution and (viii) most frequent types of errors. This information is stored in the user accounts created by each student using the system. The first four attributes are obtained at the time of user's account creation, and the others are captured while using the system.

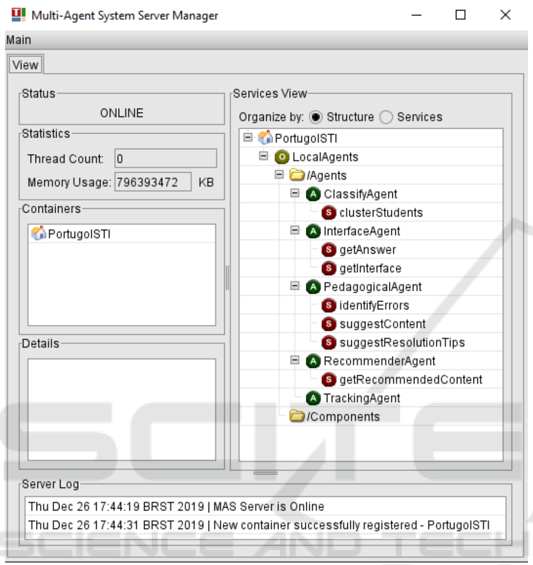
Referring Agents are responsible for recommending content to a student. It receives from the Pedagogical Agent the student group to which the student belongs and uses this group to find the content that had the best results in terms of time and user errors. When the analysis and selection is completed, it sends to the Pedagogical Agent the content that best fits your profile and level of knowledge.

This communication between agents can be represented in a BPMN model (Küster, 2012). The BPMN (Business Process Modelling Notation) is a leading process modelling language that was designed to be readable by all its business users (OMG, 2006). Figure 3 presents an example of communication between system agents using the BPMN model.

The initial process with the user requesting the pedagogical agent and then requesting the interest group that student is to the classifying agent. This agent will identify the group that the student requesting belongs to and will return to the pedagogical one. Then the pedagogical agent will send the recommending agent the group of students requesting the exercise recommendations for him. The recommender will compare content paths among students in the group and indicate the most common content among them for the student who requested it. This list returns to the pedagogical agent who responds to the student with the list of contents recommended to him, along with the exercises of the level the student is at.

4 IMPLEMENTATIONS

An implementation of the system is being developed for this work. The system is already structured on the MIDAS platform with all agents and their services. Agents on the platform interface show only communication services, which allow collaboration between agents. Responsibilities that do not require communication with other agents such as user tracking through the interface or calculating user time when using some content are intrinsic to the platform interface as can be seen in Figure 6.



Multi-Agent System Server Manager

Main

View

Status: ONLINE

Statistics
Thread Count: 0
Memory Usage: 796393472 KB

Containers: PortugolSTI

Details

Services View
Organize by: [✓ Structure] Services

```
PortugolSTI
  LocalAgents
    /Agents
      ClassifyAgent
        clusterStudents
      InterfaceAgent
        getAnswer
        getInterface
      PedagogicalAgent
        identifyErrors
        suggestContent
        suggestResolutionTips
      RecommenderAgent
        getRecommendedContent
      TrackingAgent
    /Components
```

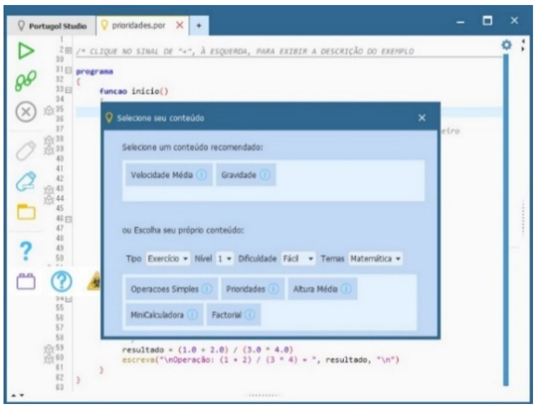
Server Log

```
Thu Dec 26 17:44:19 BRST 2019 | MAS Server is Online
Thu Dec 26 17:44:31 BRST 2019 | New container successfully registered - PortugolSTI
```

Figure 6: Agent structure modelled on MIDAS platform.

The life cycle of each agent differs according to its objectives and services. The tracking agent must capture and organize the user's information while using the system. The interface agent must communicate with the user taking questions and with the administrator presenting information. The tutor agent must select the teaching strategies for the user, using the sociability between the agents of the system. The classification agent must group similar students to facilitate the identification of characteristics. Finally, the recommendation agent must select the most adapted content according to the group of student characteristics.

The system in the Portugol interface will communicate with agents on the web platform. Figure 7 shows the system content selection image. It is under development and it will be changed, but it can already be seen inside the Portugol Studio system.



```
/* CLIQUE NO SINAL DE "+", À ESQUERDA, PARA EXIBIR A DESCRIÇÃO DO EXEMPLO
programa
{
    funcao inicio()
        resultado = (1.0 + 2.0) / (3.0 * 4.0)
        escreva("\nOperação: (1 + 2) / (3 * 4) = ", resultado, "\n")
    }
}
```

Figure 7: Interface prototype working on PS.

The above presented interface is in Portuguese idiom. It contains, on the top panel, the contents recommended by the system for the user. In the panel below, the contents that can be selected by the user are They can be filtered by: (i) type, (ii) level, (iii) difficulty and (iv) themes.

4.1 Methodological Procedures

The project was planned in several phases described below:

(i) Systematic literature review to find related work and platforms to assist the implementation

(ii) Definition of the learning trails structure

(iii) Elaboration of the system architecture

(iv) Specification of the role model for system agents

(v) Specification of agent services workflow

(vi) Adaptation of the Portugol Studio tool to receive the system as its plugin

(vii) Population of the database

(viii) Implementation of the agents

(ix) Development of the Graphical User Interface

(x) Experiments with students

(xi) Analysis of the experiments

Phases (i) to (iv) are concluded. Phases (vii), (viii) and (ix) are in progress. A partial implementation of the Agents and the User Interface was illustrated in figures 6 and 7 and the population of the database is currently being held. Phases (x) and (xi) will be held in the near future.

5 DISCUSSION

In trying to solve the problem of dropout, intelligent tutoring systems focused on programming teaching were researched and applied in Brazilian universities, but they use tools with the problems already reported in research with the PS. In recent updates, PS allows the development of plugins in the tool, so the possibility of an adaptive system being integrated into the tool has become palpable.

Thus, researching the works with Adaptive Systems and teaching methodologies, it was possible to develop the architecture presented in this work. The system described is based on two points: (i) Learning Paths and (ii) Multi-Agent System Architecture.

The learning path structure was developed by trying to use the constructivist and constructionist concepts developed by Piaget and Papert respectively. However, assisting the student in what activities he should do and allowing him a freedom to choose what he wants to do are almost opposite concepts. To solve this point, the literature sought the uses of learning paths.

In the literature, the concepts of learning paths are already used by some authors such as De Meo (2007) and Panagiotis (2016). With De Meo, paths are used to define the way a student must take to complete a project. The options to choose from in his work are only from existing projects. The paths and topics the student must follow to achieve their project goal are predefined by the system. On the other hand, Panagiotis uses learning paths as suggestions for the student. The student can select a learning object within that track without necessarily having to follow it as recommended.

Works such as De Meo's only let a momentary protagonism to the student. The student, even doing a project he has chosen, must still follow predetermined instructions. Panagiotis' work already gives the student greater freedom, the intelligent tutor being just someone who suggests what he should learn. Both works, however, do not allow the student to set up their own track.

The proposed track structure for the system uses student choices to assemble the paths. Each student using the system will receive suggestions for the following paths, but he or she can assemble their own by selecting the order of content they want to watch or play. Paths are made of content that is like Panagiotis learning objects and uses tags that define the characteristics of each content. They are an expansion of the work of Santos *el al* (2013). Not only are exercise topics, taxonomy and complexity highlighted, but also the topics they cover in explaining the topics. This allows the student to learn according to the subject of interest.

Recommend content according to the student's topic of interest and interactions with other users is part of the smart tutor's responsibilities. Modeling the smart tutor is an important part of the development process and one of the contributions of this work.

Many works on intelligent and adaptive tutors are developed with multi-agent system technologies as already mentioned in section 1. As these systems are constantly updated, they need to be modular to reduce errors and facilitate maintenance. Multi-agent systems are great for that.

In this paper, we use the concept of service oriented intelligent agents as proposed through MIDAS. Analyzing some works in the literature it was possible to see which agents are well used and which are adaptable to this work.

The use of recommending agents is an existing practice among intelligent tutors, as in the study by Cabada et al (2017). He proposes an architecture composed by a tutor and an exercise recommender. In the proposed model, the tutor is used solely to assist the user in activities with suggestions, while the recommender only provides exercises based on the user's characteristics.

This separation is applied in the proposed architecture. The recommending agent has only the function of recommending the exercises based on the characteristics of the students who have already used the system. However, the discovery of these similar characteristics is the responsibility of another agent, the classifying agent.

Both agents collaborate with the main agent, the pedagogical tutor. It coordinates agent collaboration and uses existing exercise correction modules in Portugol Studio to deliver exercise resolution tips.

This separation of intelligence from recommendation and activity aid is like that of Cabada's work. Meanwhile two other agents were added. The tracking agent, whose primary function is to track all user interactions and store them, and the interface agent that is responsible for communicating interfaces with the administrator.

Both improve system maintenance. The tracker further modularizes tasks, and the interface agent makes it easy for the administrator to change system settings without having to make changes to the source code. In addition, the interface agent allows the administrator to answer student questions and save them so that they can answer similar questions from other students.

These system features and Portugol Studio functionality can therefore help students improve their programming learning with the support of the intelligent tutor.

The tutor architecture presented here follows current technological and methodological standards. The main difference from similar work is the learning paths structure. It allows different and personal learning paths to be suggested for new students according to previous students with similar characteristics.

This might reduce the slope of the students learning curve since successful cases will be used to help them. The Portugol Studio (PS) tool also helps in the same direction since it was developed to aid novice programming students in universities, and it is widely used in Brazil.

The plugins the PS have are easily available on its interface. This will allow the proposed Adaptive System to be found by many Computer Science students in Brazil and also independent learners and will certainly foster the development of new research projects.

6 CONCLUSION AND FUTURE WORKS

There is a recurring need in the Computer Science area for tools to assist the learning process, especially programming logic. Students often do not have time to study or cannot adapt to classroom teaching. Adaptive systems emerge as one of the alternatives to this reality.

Alternatively, constructivist and constructivism methodologies can also help students learn new concepts in programming using IDEs. Systems with adaptive features and assisted learning techniques like Portugol Studio can be a way to facilitate learning for beginning programmers.

The main contribution of this work is the proposal of an architectural model with low-level diagrams representing the functionalities of the agents for an adaptive system. The purpose is to facilitate the logic learning and programming with learning paths. This can generate gains in preparing a generation in a high demand market for this knowledge type.

The next steps in implementing the system are as follows: (i) implementation of the services, (ii) population of the content tags and (iii) implementation of the communication interface between Portugol and the Agents on the web.

The implementation of the services will allow communication between agents on the MIDAS platform. Each agent service implemented must follow the definitions shown in subsection 3.3.

Content tags are one of the bases of the system's intelligence. They will allow the recommending agent to identify topics of interest to students in each content. This will permit him to learn to program within his context of interest, following constructionist theory.

Finally, Portugol's communication interface with web agents will start the connections that will allow agents to receive student information and collaborate to send students learning suggestions.

As the system is finished, the future works in the project are: (i) experiments with students and (ii) analysis of the experiments, following the procedures described on subsection 4.1.

The experiments will be held made with classes of computer science real students of the authors from the local university. The students will use the system as a support to discover their difficulties in programming.

The main goal is to assess whether the system gives them coherent personalized suggestions according to their difficulties and preferences. The student's acceptance on the recommendations are also going to be measured.

Since the Portugol Studio tool is widely adopted in Brazilian universities, further work also can be done to collected data to find patterns about general difficulties of Brazilian programming learners and which learning paths can help the most.

These researches might help find better ways to help students learn programming and reduce the dropouts on universities caused by the difficulties they have in the initial courses.

REFERENCES

Bada, S. O., & Olusegun, S. (2015). Constructivism learning theory: A paradigm for teaching and learning. *Journal of Research & Method in Education, 5(6), 66-70.*

Baranauskas, M. C. C., Vieira, H., Martins, R. M. C., & D'ABREU, J. V. (1999). Uma taxonomia para ambientes de aprendizado baseados no computador. O computador na sociedade do conhecimento, 45.

Cabada, R. Z., Estrada, M. L. B., Hernández, F. G., Bustillos, R. O., & Reyes-García, C. A. (2018). An affective and Web 3.0-based learning environment for a programming language. *Telematics and Informatics, 35(3), 611–628*. doi:10.1016/j.tele.2017.03.005

CareerCast (2019) The Toughest Jobs to Fill in 2019, Available at: https://www.careercast.com/jobsrated/2019-most-difficult-jobs-to-fill (Accessed: 30th December 2019).

Carmen-Leocadia Pesantez Pozo (2017) 'Smart Education and Smart e-Learning', *Conference proceedings of »eLearning and Software for Education« (eLSE), III(01), pp. 89-95*.

Cidral, W. A., Oliveira, T., Di Felice, M., & Aparicio, M. (2018). E-learning success determinants: Brazilian empirical study. *Computers & Education, 122, 273–290*. doi:10.1016/j.compedu.2017.12.001

Cooper, S., Dann, W., & Pausch, R. (2000). Alice: a 3-D tool for introductory programming concepts. *Journal of Computing Sciences in Colleges, 15(5), 107-116*.

De Meo, P., Garro, A., Terracina, G., & Ursino, D. (2007). Personalizing learning programs with X-Learn, an XML-based, "user-device" adaptive multi-agent system. *Information Sciences, 177(8), 1729–1770*. doi:10.1016/j.ins.2006.10.005

Dolenc, K., & Aberšek, B. (2015) "TECH8 intelligent and adaptive e-learning system: Integration into technology and science classrooms in lower secondary schools" *Computers and Education, 82, 354–365*, doi:10.1016/ j.compedu. 2014.12.010

Elham Mousavinasab, Nahid Zarifsanaiey, Sharareh R. Niakan Kalhori, Mahnaz Rakhshan, Leila Keikha & Marjan Ghazi Saeedi (2018) Intelligent tutoring systems: a systematic review of characteristics, applications, and evaluation methods, *Interactive Learning Environments*, doi: 10.1080/10494820.2018. 1558257

Frade, R. V. C. (2015) "UNIVIRTUAL – Ambiente Virtual 3D Multiagente com Recomendação Personalizada de Objetos de Aprendizagem", Dissertação (Mestrado em Ciência da Computação) - *Universidade Estadual do Rio Grande do Norte, Universidade Federal Rural do Semi-Árido, Mossoró*.

Giraffa, L. M. (1999). Uma arquitetura de tutor utilizando estados mentais. 1999 (Doctoral dissertation, Tese (Doutorado em Ciências da Computação) – *Instituto de Informática, UFRGS, Porto Alegre)*.

Giraffa, L. M., & da costa Mora, M. (2013). Evasão na disciplina de algoritmo e programação: um estudo a partir dos fatores intervenientes na perspectiva do estudante. In *Congresos CLABES*.

Gonçalves, E. J. T. (2009). Modelagem de arquiteturas internas de agentes de software utilizando a linguagem MAS-ML 2.0 (Doctoral dissertation, Dissertação de Mestrado. Universidade Estadual do Ceará. Centro de Ciência e Tecnologia. Fortaleza).

Guia da Carreira (2018) Saiba quais as 8 profissões que mais crescem no Brasil, Available at: https://www.guiadacarreira.com.br/profissao/profissoes-que-mais-crescem/ (Accessed: 30th December 2019).

Haendchen Filho, A. (2005). Um Framework do tipo Middleware para Sistemas Multi-Agentes na Internet (Doctoral dissertation, PUC-Rio).

Filho A., Thalheimer J., Dazzi R., Santos V. and Koehntopp P. (2019). Improving Decision-making in Virtual Learning Environments using a Tracing Tutor Agent. In *Proceedings of the 21st International Conference on Enterprise Information Systems - Volume 1: ICEIS*, ISBN 978-989-758-372-8, pages 600-607. DOI: 10.5220/0007744006000607

Harley, J. M., Bouchet, F., Hussain, M. S., Azevedo, R., & Calvo, R. (2015) "A multi-componential analysis of emotions during complex learning with an intelligent multi-agent system". *Computers in Human Behavior, 48, 615–625*, doi:10.1016/j. chb.2015.02.013

Hodecker, Andrei. Aprimoramento e avaliação do corretor de questões do Portugol Studio. 2014. TCC (graduação em Ciência da Computação) - Universidade do Vale do Itajaí, Itajaí, 2014.

Hooshyar, D., Ahmad, R. B., Yousefi, M., Yusop, F. D., & Horng, S.-J. (2015) "A flowchart-based intelligent tutoring system for improving problem-solving skills of novice programmers", *Journal of Computer Assisted Learning, 31(4), 345–361*.

Küster, T., Lützenberger, M., Heßler, A., & Hirsch, B. (2012). Integrating process modelling into multi-agent system engineering. *Multiagent and Grid Systems, 8(1)*, 105–124. doi:10.3233/mgs-2012-0182

Lobo, R. (2017) 'A Evasão No Ensino Superior Brasileiro – Novos Dados', Estadão, 07 October.

MichaelPage (2019) The World's Most In Demand Professions, Available at: https://www.michaelpage.co.uk/minisite/most-in-demand-professions/ (Accessed: 30th December 2019).

Ng, S. C., Choy, S. O., Kwan, R., & Chan, S. F. (2005). A Web-Based Environment to Improve Teaching and Learning of Computer Programming in Distance Education. *Lecture Notes in Computer Science, 279–290*. doi:10.1007/11528043_28

Noschang, L. F., Pelz, F., & Raabe, A. (2014). Portugol studio: Uma ide para iniciantes em programaçao. Anais do CSBC/WEI, 535-545.

Nwana, H. S. (1990). Intelligent tutoring systems: an overview. *Artificial Intelligence Review, 4(4), 251-277*.

Oliveira, C. M., Pimentel, A., & Krynski, E. M. (2015, October). Estudo sobre o sequenciamento inteligente e adaptativo de enunciados em programaçao de computadores. In *Anais dos Workshops do Congresso Brasileiro de Informática na Educação (Vol. 4, No. 1, p. 1320)*.

OMG Business Process Modeling Notation. "Version 1.0." OMG Final Adopted Specification. *OMG (2006)*.

Paiva, J. C., Leal, J. P., & Queirós, R. A. (2016). Enki. *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education - ITiCSE '16*. doi:10.1145/2899415.2899441

Papert, Seymour. "A máquina das crianças." *Porto Alegre: Artmed (1994)*.

Panagiotis, S., Ioannis, P., Christos, G., & Achilles, K. (2016). APLe: Agents for Personalized Learning in Distance Learning. *Computer Supported Education, 37–56*. doi:10.1007/978-3-319-29585-5_3

Pattabiraman K. (2019) LinkedIn's Most Promising Jobs of 2019, Available at: https://blog.linkedin.com/2019/january/10/linkedins-most-promising-jobs-of-2019 (Accessed: 30th December 2019).

Pelz, Fillipi Domingos. Correção automática de algoritmos no ensino introdutório de programação. 2011. TCC (graduação em Ciência da Computação) - *Universidade do Vale do Itajaí, Itajaí, 2011*.

Philips, D. C. (1995). The Good, the Bad, and the Ugly. The many Faces of Constructivism. I.

Phobun, P., & Vicheanpanya, J. (2010). Adaptive intelligent tutoring systems for e-learning systems. *Procedia - Social and Behavioral Sciences, 2(2), 4064–4069*. doi:10.1016/j.sbspro.2010.03.641

Piaget, J. (1980). The psychogenesis of knowledge and its epistemological significance. In *M. Piatelli-Palmarini (Ed.), Language and learning (pp. 23-34). Cambridge, MA: Harvard University Press*.

Raabe, A., Zanini, A. S., Santana, A. L. M., & Vieira, M. F. V. (2016). Influência dos enunciados na resolução de problemas de programação introdutória. *Revista Brasileira de Informática na Educação, 24(1), 66*.

Raabe, A. L. A., & Silva, J. D. (2005). Um ambiente para atendimento as dificuldades de aprendizagem de algoritmos. In *XIII Workshop de Educação em Computação (WEI'2005). São Leopoldo, RS, Brasil (Vol. 3, p. 5). sn*.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., ... & Kafai, Y. B. (2009). Scratch: Programming for all. *Commun. Acm, 52(11), 60-67*.

Romagosa i Carrasquer B. (2019) The Snap! Programming System. In: *Tatnall A. (eds) Encyclopedia of Education and Information Technologies. Springer, Cham*.

Santos, A., Gomes, A., & Mendes, A. (2013). A taxonomy of exercises to support individual learning paths in initial programming learning. *2013 IEEE Frontiers in Education Conference (FIE)*. doi:10.1109/fie.2013.6684794

Silva Filho, R. L. L., Motejunas, P. R., Hipólito, O., & Lobo, M. B. C. M. (2007). A evasão no ensino superior brasileiro. *Cadernos de pesquisa, 37(132), 641-659*.

Tarus, J. K., Niu, Z., & Mustafa, G. (2017). Knowledge-based recommendation: a review of ontology-based recommender systems for e-learning. *Artificial Intelligence Review, 50(1), 21–48*. doi:10.1007/s10462-017-9539-5

Trade Schools (2019) 31 High-Demand Jobs in 2019 for Almost Every Type of Person, Available at: https://www.trade-schools.net/articles/high-demand-jobs.asp (Accessed: 30th December 2019).

Vaidya, N. M., & Sajja, P. S. (2016). Agent based system for collaborative learning environment in an educational habitat. *2016 International Conference on ICT in Business Industry & Government (ICTBIG)*. doi:10.1109/ictbig.2016.7892644

Weragama, D. S. (2013). Intelligent tutoring system for learning PHP (Doctoral dissertation, Queensland University of Technology).

Wolber, D. (2011). App inventor and real-world motivation. *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education - SIGCSE'11*. doi:10.1145/1953163.1953329

Yaghmaie, M., & Bahreininejad, A. (2011). A context-aware adaptive learning system using agents. *Expert Systems with Applications, 38(4), 3280-3286*.