ASSE, Simposio Argentino de Ingeniería de Software

AgileTalk: A Conversational Virtual Reality Synthetic Platform for Training in Agile Processes

Marcelo Campo, Analía Amandi, Nelson Acosta, Matías Guerrero

{mcampo,amandi,nacosta, megerrero}@exa.unicen.edu.ar

NICE Research Group, Faculty of Exact Sciences, UNICEN University

Tandil, Bs. As., Argentina

Abstract

In this article, the main motivations and conceptual basis of using chatbot-based technology as the platform to support education and training on agile development are presented, using ScrumTalk as an example. ScrumTalk is a prototype of an intelligent conversational core for the AgileTalk platform under development for an Intelligent Synthetic Agile Software Development Environment. AgileTalk represents an ambitious evolution of Virtual Scrum (a Virtual Reality-based platform for Scrum training we developed some years ago) towards a synthetic 3D conversational-based full-fledged process training and development environment. This evolution means a step further "Virtual" environments towards "Synthetic" ones where actors can be whatever human or synthetic alter egos that interact through natural language to carry out the activities involved in a defined process according to their role. ScrumTalk represents the proof of concept for the AgileTalk platform's underlying paradigm.

1. Motivation and Background

Agile methods enable companies that implement them to address environmental unpredictability by deploying strict control mechanisms and a strong emphasis on iterative processes. But, teamwork self-organization represents the most challenging change for any organization.

This strong emphasis on teamwork naturally led to a high dependency on human skills that require experience and constant training. In this context, the exercise of apprentices plays a crucial role in Agile organizations. Moreover, as a philosophical consequence, an agile approach to the training process appears as the most logical approach to fulfill the business goals.

For the past ten years, the group has been exploring Artificial Intelligence and Machine Learning techniques in virtual reality environments for agile training of Software Engineers for agile environments. This work led to the evolution of Virtual Scrum [18], a platform that uses a virtual world to simulate a natural work environment handling 3D displays of the Scrum artifacts. Virtual Scrum also supports a task board for planning and tracking user stories; a daily meeting artifact for solving problems, adapting rapidly to changes, and removing impediments; and a burn-down chart to reflect on the past sprint and make continuous process improvements during retrospective meetings. This way, Virtual Scrum uses a virtual world to harness the 3D interfaces for training students in their performance in a simulated Scrum environment. Fig. 1 shows several snapshots of the activities supported by VS in real teaching projects carried out since 2013. .Additionally, from a pedagogical point of view, we explored the concept of Felder-Silverman Learning Styles [1] as an intelligent boost for personalizing training strategies based on individual psych-pedagogical preferences. Realizing the results of the approach, we went further until reaching the ability of automated detection of soft-skills in scrum-based software development activities [2].

Virtual Scrum was the first attempt to advance in agile virtual training for agile processes, and the subject has gained a growing interest after its inception. However, despite the results achieved in this exploratory work, VirtualScrum was a passive support tool limited to give some relatively naïve suggestions reactively. So, to boost all its potential, we advance, taking advantage of the power of today's NLP support, envisioning a *proactive* platform: *AgileTalk*.

AgileTalk is based on the idea of *AgileBots*, intelligent agents that act as synthetic alter egos of human actors involved in agile processes interacting through the natural language process support provided by the RASA

open-source framework [17] and the VR Engine Unity. This approach aims for a platform for experimenting in a broad spectrum of applications from complex simulations for training up to concrete process-centered development environments taking into account' psych-pedagogical models.

Following an agile iterative and incremental process, *ScrumTalk* was developed as a proof of concept. *ScrumTalk* is a prototype for experimenting with different Scrum-related ceremonies emphasizing the transparent interaction between human and synthetic actors.

This article introduces the foundations of *AgileTalk* through an example of *ScrumTalk*, organized into three related parts. First, it outlines the conceptual framework behind *AgileTalk* and the *AgileBot* concept. The second part is centered on describing its fundamentals through an example of a specific instantiation of the RASA framework for supporting *ScrumTalk.* Finally, the more advanced features of psycho-pedagogical detection and simulation under research are briefly described, along with preliminary conclusions and future work.

2. *AgileTalk*: From *Chatbots* to *AgileBots*

AgileTalk goes beyond the original extent of a simple chatbot as an interface agent towards *AgileBots*, agents that interact indistinctly among them or humans in natural language. It can be seen as a multi-agent system with the essential difference that all the communication among *AgileBots* (agents) is based on natural language. This kind of interaction (not too efficient indeed but realistic instead) allows for the interaction with either his Team Member owner, other Team Members, or even other *AgileBots* in a human-like conversation.

Because of this, we refer to such environment as "Synthetic" instead of "Virtual" just because it is not virtual. Still, it exists as the natural place where a process is enacted indistinctly and transparently by humans or *AgileBots*.

Acting as a synthetic alter ego of a team member of an agile process, an *AgileBot* holds any valuable data and basic capabilities that its human counterpart should contribute to the project.

Fig. 1depicts a conceptual view of an *AgileBot* as divided into two *metaphorical hemispheres*, the Action and the Profile ones. The Action Hemisphere refers to all the behavior related to the bot's functions as an assistant/representative of the Team member while working on the project. These behaviors may include acting as a *Development Assistant* in development-related tasks or as a limited but valuable *Representant.* In the first role, the actions can consist of working as a communication proxy with other team members, recalling activities and meetings and other project-related activities, execute any configured action related to the associated artifacts according to the defined process, etc. As Representant it could autonomously inform advances and states of ongoing or done tasks, to participate in a limited but valuable way in project meetings and reviews or at management roles autonomously control of plans evolution and delays, among others

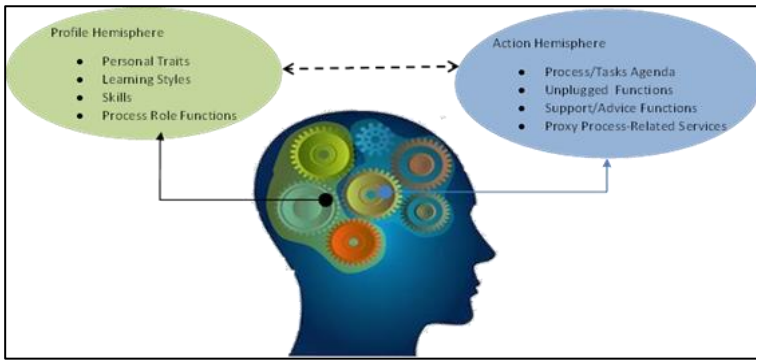

Fig. 1 AgileBot's conceptual scheme functions

The Profile Hemisphere refers to all the data associated with personality traits and learning styles collected through interviews and skills inferred by observing and analyzing the Team Member's behavior during the work. This distinctive feature, inherited from Virtual Scrum [21] is complemented with MBTI [14] and B5 [16][9] profiles[1]. These profiles together enable the development of realistic project simulations as well as

[1] It must be noticed that, at the current stage of the development, MBTI and B5 profiles are not automatically obtained but trough the formal written tests applied to participants of experiments. This aspect is currently under research as a central part of the project.

the intelligent analysis and support for essential agile ceremonies, for example: Help the Team Manager in the semi-supervised automation of agile development teams, like generating responses to requests according to the Team Member Personality, creating customized training scenarios, explore alternative strategies for team management and control that can be built on top of this information, personality-based group performance simulations, etc.

2.1 Conceptual Architecture View

Any software development process can be described through the interactions among three basic generic entities, namely Actors, Activities, and Artifacts. Actors play different roles related to one or several Activities that create and transform Artifacts. Artifacts are, essentially, pieces of information expressed as structured, semi-structured, or non-structured interrelated documents held in a shared repository and manipulated through standard external tools for team-based software development.

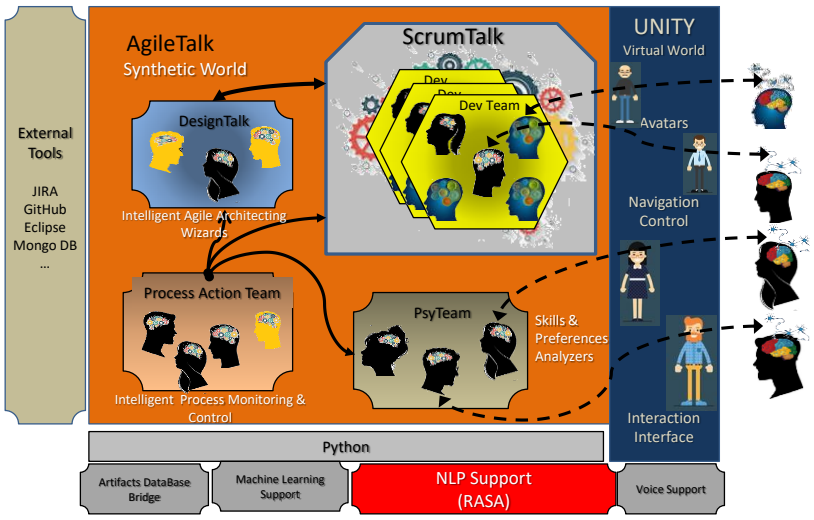

Fig. 2 Conceptual architecture view of AgileTalk

The agile generic process support provided by *AgileTalk* is inspired by the Scaled Agile Framework (SAFe ™) and the CMMI integration framework, enforcing the idea of Agile Architecting. All these activities are carried under the Synthetic metaphor where *TeamBots* play active and passive roles. Therefore, specialized *TeamBots* Wizards, hereafter *DesignBots*, are in charge of building and evolving the final architecture built by agile teams. In addition, the building process is monitored and controlled by the Action Process *TeamBots*.

Fig. 2 present a conceptual view of the AgileTalk architecture. The core is constituted by *AgileBots* playing different roles within a process as their human owners play:

- ***TeamBots***: AgileBots devoted to Agile Project Development
- ***DesignBots***: AgileBots specialized in intelligent aided agile architecting. A chatbot-centric evolution of the *DesignBot* environment [5]
- ***PATBots***: AgilesBots implementing the agile integration of CMMI practices for Process Monitoring and Control [4]. This Synthetic department is in charge of guiding and monitoring the different process actors according to the defined process for a project.
- ***PsyBots***: Profiler team for detection and analysis of learning styles and personal skills used to guide the behavior and responses of personalized *TeamBots*. These AgileBots are in charge of providing support for the personalized intelligent behavior expected for each TeamBot, acting as a Representative of each Team Member.

From an implementation point of view, the *AgileTalk* core heavily relies on the RASA [17] framework for supporting natural language interaction (among humans or *AgileBots*) and the powerful visual support provided by the VR Unity platform.

2.2 RASA: NLP Support and Beyond

RASA is a robust framework that provides all the necessary support to construct a conversational interface in natural language and tailor it through Python. It is based on the notion of "domain". A domain defines the universe in which a chatbot operates. It specifies the intents, entities, slots, responses, forms, and actions that the bot should know. Settings are also defined for conversation sessions, such as the conversation time [17] presents a control view of RASA architecture in UCM notation. The end-user asks the chatbot, so the Rasa NLU (Interpreter) structures the message into an output containing the original text, the intent, and the entities involved. The Tracker is in charge of maintaining the conversation state and process the structured output from the interpreter. The Policy acts on the current state of Tracker, deciding which following appropriate action is to be performed. Finally, the log of the selected *action* is sent to the Tracker, and an appropriate response is provided to the user, using intents defined as an utter response.

2.3 Intention Detection: The Hinge Mechanism

The automatic detection of the "intention" behind a verbal message is a core concept in RASA. It also is the core of *AgileTalk's* structuring mechanism. Through the typical training process of machine learning approaches, RASA can build an NLP classifier that can *detect intentions* in a dialog with few examples. These *intentions* can vary from a basic model on interpreting different forms of greeting phrases up to much more complex expressions that hold either a literal meaning or the search for a potential solution to a given problem.

This feature represents the key for mapping expressions in natural language to specific events that guide the architectural flow within computational components. For example, Fig. 4 shows a RASA specification for a simple hypothetical dialog explaining a Kanban board. Let's suppose that the apprentice is a centennial. So a question in their dialect usually could be "*I foot on the client, and everything broke down*," trying to express that "*I modified the client variable and the function gave unexpected results*…". To *AgileTalk*, this phrase could mean the intent "start the debug action" and trigger an event that activates the code revision task. So, if this intent is trained with the first example, it will trigger the "code revision" event.

Furthermore, this mechanism enables the design of transparent either *TeamBot-TeamBot* or *Human-TeamBot* communication. Therefore, beyond the natural application to conversational interfaces, RASA provides a robust set of abstractions that can be mapped easily to assimilate stereotypes of the software development process. This aspect represents the most novelty approach taken in *AgileTalk* and the reason for its name.

3. *ScrumTalk*: The proof of concept

These ideas are the conceptual basis of the project ScrumTalk, a proactive, intelligent conversational 3D environment we started to explore as a proof of concept. *ScrumTalk* is an operational prototype that enables simulations beyond the typical virtual games for just teaching Scrum-related activities [10].

Instead, *ScrumTalk's* aim is centered on an augmented process enacting model. Under this ideal model, apprentices or trainees could interact immersed in a mixed reality project already executed in the organization with *TeamBots* performing the activities carried out by Team Members in real projects. This interaction is developed in natural language, so human Team Members playing different development roles can be intermixed transparently in the process. This approach serves the purpose of teaching typical ceremonies, but it enables the detection of the trainee's technical and soft skills using several current techniques for data analysis.

3.1 A simple example: The classical HR assistant receiving a novice future *scrummer*

Fig. 3 presents an example of the different capabilities of ScrumTalk seen from the supervisor's point of view. The sequence shows various stages of a guided tour through the #MicroDesign company's Scrum process, where Claudette (a TeamBot (playing the HR assistant role) guides the human aspirant Matheu under the supervisor's watch.

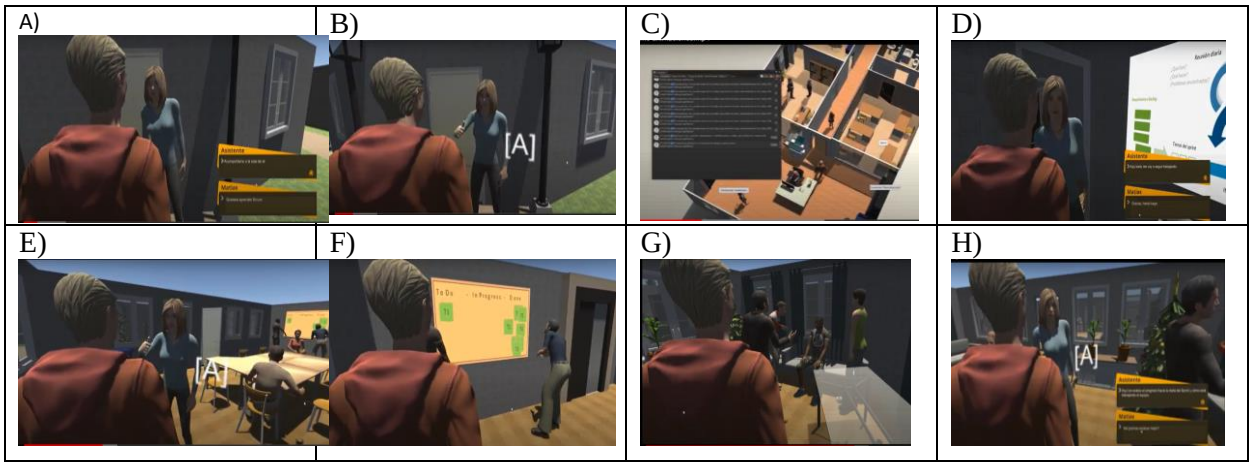

Fig. 3 Movie-mode sequence

This sequence presents scenes from the reception, explaining the company's approach ending withholding daily meetings among the team members. Fig. 4 A -B shows a typical reception where Claudette (the TeamBot) greets Mathew (the human candidate) and invites him to follow her to see the process in action. In Fig. 3 C the supervisor is watching their way into the building and can intervene at any moment to help the TeamBot with some explanation that it can't manage through the voice interface. At the same time, the *TeamBot* records the textual translation of the speech for further training for the story (somehow a way of supervised learning).

All these interactions are implemented in RASA as *stories* or conversation flows. For example, Fig. 4 corresponds to the moment of the conversation where the apprentice is instructed about using the Kanban board (see Fig. 3 E - F). Therefore, a relatively trivial "kanban" story can be defined, making use of the set of intentions set as "Kanban board" ("What is a kanban board?") or "Kanban tasks" ("How are the tasks placed?"). In other words, through them, the message purpose about knowing the use and distribution of the tasks within the board is detected ending in the response from the TeamBot, using actions such as "utter_kanban1" ("It is the tool to map and visualize the workflow").

```
        // Traing dataset
intent: basicExplanation examples:
      Why you speak about a board?
      What is kanban?
intent: queesKanban examples: 1
      Can you explain to me about the kanban?
      Can you explain how the board works?
intent: kanbanTasks examples:
        How are tasks placed?
```

```
        // Dialogflow description
story: kanban steps:
      intent: basicExplanation
      action: utter kanbanl
      action: utter kanban2
      intent: kanbanTasks
      action: utter kanbanTasks
      intent:   agradeceExp
      action: utter_coninueWorking
```

```
utter kanbanl:
   text: " It is the tool to map and visualize the workflow."
utter kanbanTasks:
      text: " They are placed on a kanban board. Accompany me, there is an employee who can explain you better "
```

Fig. 4 Sample of a basic RASA dialog specification

In this way, the model implemented in RASA associates stories defined for each particular situation, which are then mapped to the dialogue flows according to the roles of the virtual members within the process. This mapping allows the composition of very complex functionality depending on the *AgileBot's* position.

3.2 Modeling multiple roles

Stories allow modeling the dialogue and intelligence of each *TeamBot* according to the role they can play in a process flow. For example, when an intention is detected, dynamic responses can be generated according to their rank within Scrum, either to utter a predefined reaction or a custom action (implemented via Python). Fig. 5 shows an example of the modeling of *TeamBots* playing different roles within a process flow. For example, when the Sprint Backlog is being carried out and later a Daily Scrum, a Scrum Master will intervene. In this way, it is

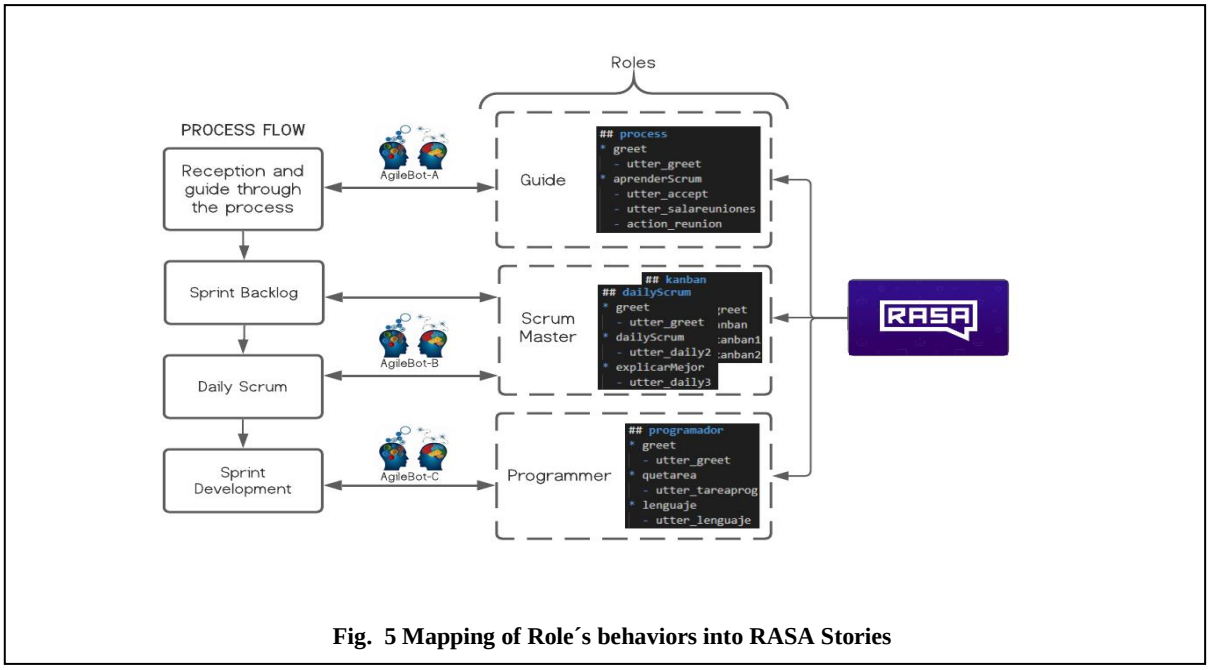

Fig. 5 Mapping of Role´s behaviors into RASA Stories

possible to simulate a work environment with *TeamBots* capable of assuming positions within the agile process, which guarantees to develop each activity similar to the business environment. Through the intelligence developed, *TeamBots* take their functions; interact both among them and with the rest of the Team Members.

3.3 Modeling Ceremonies

Fig. 6 represents another possible flow of dialogue associated with the *dailyScrum* story in Fig. 3 G-H. Here the apprentice intends to understand the ceremony asking the assistant for a more detailed explanation. This question is mapped to the *"explainMore*" intent triggering the custom action *ActionMeeting* of them through the intention "*dailyScrum*".

The example shows using the custom actions mechanism supported by RASA as a hotspot to be implemented in Python. The action called "action_meeting" makes use of the Tracker to access the *AgileBot's* memory as part of the custom actions, thus knowing the events that have occurred and the current state of the conversation (Fig. 6). In this example, then, an event that is part of the component is captured, which allows access to the last dialog issued by the bot: "Follow me to the DailyScrum meeting, please". This implementation enables to guide the apprentice through the flow of the process to be part of the process links. In this way, by detecting the user's intent, the assigned guide will indicate where to go within the virtual world.

The visible behavior results from the cooperation between the TeamBot in the Synthetic world and its corresponding Avatar into the Unity Virtual World. The Python component interacts with RASA to get a tracker object that manages the dynamic responses and location to follow the AgileBot. Through this connection, the *TeamBot* can interact with Unity generating the visual effect of the Avatar walking with the assistant *AgileBot* up to the target area of the virtual building. The path the assistant *AgileBot* takes is part of the shared knowledge held

story : dailyScrum
steps:
intent: greet
action: utter_greet
intent: dailyScrum
action: utter_daily2
intent: explainMore
action: action meeting
	class ActionMeeting (Action):
	
	def name (self) -> Text:
	return "action meeting"
	
	def run (self, dispatcher: CollectingDispatcher , tracker Tracker,domain:
	Dict [Text |, Any]) -> List[Dict[Text, Any]]:
	
	bot_event = next (e for e in reversed(tracker.events) if e ["event"]== "bot"
	text = bot_event.get ("text")
	if text = "Follow me to the DailyScrum meeting, please" :
	# Dispatch Unity Event
	dispatcher.utter_message(json_message={"Room": "DailyScrum"))
	return []

Fig. 6 A simplified Phyton custom action

in the Virtual World roadmaps, which is translated to directional commands in Unity protocols that follow the avatars.

In this trivial scenario, one of the most distinctive features of AgileTalk, the agile personalization component, comes silently into the scene.

3.4 Get Smart with *Policies*

Stories can be defined statically or dynamically. Dynamic generation of *stories* enables to adapt the dialog according to data gathered during a conversation. The *Policies* component of RASA is the hotspot to provide *intelligent behavior* by prescribing the actions taken at each step of a given conversation. This mechanism makes it possible to customize a related set of dialogues through custom actions, allowing the dialog flow's dynamic adaptation. The RASA engine will apply the Policy that returns a higher confidence level when multiple policies are used.

This mechanism is used to implement the daily meeting dialog emulated by three *TeamBots* and a Scrum Master Fig. 3 G-H, as shown in Fig. 7, through the CustomPolicy Python class. If the developer is a listener, this Policy will execute an *action_listen* waiting for new user input. Otherwise, it will provide a response or action from the developer. This response is determined by functions that return a proper answer according to the personality preferences the TeamBot embodies.

4. Human Factors: The *PsyTeam*

The most distinctive feature of AgileTalk is "watching" from the background: The *PsyTeam*. This feature is a central one that intervenes in almost all the activities a Team Member develops into *AgileTalk*. Specialized *PsyBots* monitor the behavior of *TeamBots* collecting valuable information that feeds the profile of a Team Member.

This profile includes the Learning Style (ILS) [7], along with the Myers-Brig Type Indicator (MBTI) [14] [9] and the Big Five model [16][3] personality traits. Together they give parameters of different aspects of a person's learning preferences and personality traits valuable for inferring potential behaviors in teamwork [15], preferred positions [23], and soft skills (crucial for agile) [6][12].

A Learning Style can be roughly defined as an abstraction of the preferred ways and means a student or apprentice acquires knowledge. This information is helpful for a teacher or coach to accommodate the explanations so the apprentice can learn in a faster and accurate fashion. Also, as we showed in [21], it is valuable to identify potential soft skills in agile development. The Felder-Silverman learning style model (ILS) [7] characterizes each learner according to four dimensions: <Sensing-Intuituive, Sequential-Global, Active-Reflective, Visual-Verbal> that comprises 16 learning styles.

```
Scrum Master: Good morning
Developer1: Hello.
Scrum Master: Hello Smith. What were you doing yesterday?
Developer1: I made the application's dark mode
Scrum Master: And how did it go? Did you have any problem?
Developer1: I had no problems
Scrum Master: Great, what are you going to continue with?
Developer1: I am going to finish my previous task
Scrum Master: Ok, see you tomorrow
Developer1: see you
Scrum Master: Hi buddy you look fine!, What were you doing?
Developer2: Hi Master.  I have been working on the database
Scrum Master: Do you have any trouble?
Developer 2:  Oh yea… let me tell you…
------
class CustomPolicy(Policy):

   def predict_action_probabilities(
      self,
      tracker: DialogueStateTracker,
      domain: Domain,
      interpreter: NaturalLanguageInterpreter,
      **kwargs: Any,
   ) -> PolicyPrediction:
```

```
res = self._default_predictions(domain)
sender_id = tracker.current_state()['sender_id']

if (sender_id != Listener):
   if(not self.answered):
      if (intent == 'doYouHaveProblem'):

      # Dinamically builds the action name depending on
       # personality traits
            prox = action_type + properAnswer(sender)
             res= confidence_scores_for(prox,1.0,domain)
          else:
             res = confidence_scores_for(ans, 1.0, domain)
          self.answered = True
        else:
           self.answered = False
     else:
        result = confidence_scores_for('action_listen', 1.0,
domain)
        self.answered = True
….
# Generated RASA responses
utter_check_previous_status_extraverted:
text : "Hi [treatment], what were you doing?"

utter_check_previous_s tatus_respectful :
text :  "Hello {name}  :  What  were  you  doing
yesterday?
```

Fig. 7 Custom Policy Brief Example

Myers and Briggs developed an indicator of personality types adapting from Carl Jung's type theory. It is based on 16 personality types, which act as a reference, analyzed through four dimensions that became very important in group theory. A four dimension vector describes an MBTI profile: <Extraverted-Introverted, Sensitive-Intuitive, Thinking-Feeling, Judging-Perceiving>. The values in each dimension determine a personality type with different degrees of preferences in one of the two possible orientations in each axis. These concrete values lead to the definition of patterns of potential reactions.

The Big Five is an alternative personality model that considers five fundamental dimensions (divided into two sub-dimensions) for describing and evaluating personality traits, <Extraversion, Affability, Consciousness, Neuroticism, Openness>, according to [3][2]. As the names suggest, these dimensions give indicators of potential behaviors of a Team Member in interpersonal interactions crucial to agile teams.

These models are usually used to describe stereotypes either of personality or aptitudes preferences. These stereotypes can be used as patterns of behavior and aptitude for managing situations. When understood, they are a vehicle to describe an approximate estimation of reactions and capabilities that make it feasible to model and simulate them.

Together, defining a generic profile for experimenting conform to *AgileTalk*'s profile model basis. The personality stereotypes are used to emulate the behavior of a *TeamBot* representing its human owner in meetings and the preferred treatment a *TeamBot* should give to others, as will be shown below. Indeed, in the end, this model is intended for HR experts experimenting with a potential combination of *TeamBot*'s behaviors that can be used to generate development teams satisfying some interesting criteria using machine learning techniques.

These hypotheses are encouraged by results obtained in an ongoing research project on agile training since several years ago. Fig. 8 present an excerpt of one actual agile course project with the joint profiles of the

[2] There are several variants proposed in the literature that consider different sub-dimensions or even dimensions. The model used in this work is an attempt to bring some uniformity to common concepts.

Artemisas team and the MBTI role preferences according to [23]14. Superscripts represent the qualitative or numeric value in each dimension established by the three models of each apprentice, indicating the level of preference in each dimension. The first column shows the affinity cluster number obtained through the X-means algorithm. The algorithm is trained with characteristic vectors defined according to an equilibrated number of apprentices satisfying the MBTI role preferences combined with Extraversion, Consciousness, and Neuroticism as the main *trait drivers*. Color emphasis is used to highlight the traits drivers selected for grouping.[3]

	Artemisas			
	ILS	MBTI	B5	Apprentice
3	$S^{mh}V^{mh}R^{ml}G^{m}$	$E^{lo}S^{xa}T^{ml}J^{lh}$	$E^{96}A^{81}T^{81}N^{48}M^{84}D^{35}$	
	$I^{ev}V^{ml}A^{ev}Q^{hl}$	$E^{lh}S^{lh}F^{vl}J^{ml}$	$E^{62}A^{88}T^{83}N^{89}M^{75}D^{38}$	
3	$S^{hl}V^{hl}R^{ev}G^{ev}$	$E^{lo}S^{ml}F^{lo}P^{lo}$	$E^{72}A^{103}T^{102}N^{88}M^{75}D^{41}$	
4	$S^{mh}V^{hl}A^{hl}Q^{ev}$	$E^{lo}S^{vl}F^{lo}P^{ml}$	$E^{72}A^{94}T^{68}N^{81}M^{77}D^{33}$	
3	$S^{ml}V^{mh}A^{ml}G^{ml}$	$E^{ml}N^{lh}F^{lh}P^{lo}$	$E^{88}A^{88}T^{98}N^{32}M^{93}D^{50}$	
4	$S^{hl}V^{hh}A^{hl}G^{ev}$	$E^{xa}N^{lh}T^{lo}P^{lh}$	$E^{87}A^{76}T^{82}N^{72}M^{63}D^{25}$	
4	$I^{mh}O^{ml}A^{ml}Q^{ml}$	$E^{xa}N^{xr}T^{lo}J^{ml}$	$E^{73}A^{70}T^{79}N^{63}M^{70}D^{32}$	
1	$S^{hh}V^{hl}A^{ml}Q^{ml}$	$E^{xa}S^{lh}T^{vh}J^{lh}$	$E^{89}A^{86}T^{95}N^{95}M^{86}D^{38}$	

	Common Preferences				Programmers				Designers			
	ST	SF	NF	NT	ST	SF	NF	NT	ST	SF	NF	NT
IJ	3	1	1	1	3	1	1	1	3	1	1	1
IP	3	1	3	1	3	1	3	1	3	1	3	1
EP	0	2	1	2	0	2	1	2	0	2	1	2
EJ	2	2	0	1	2	2	0	1	2	2	0	1
	8	6	5	5	8	6		5	8	6	5	5
1				9				19				24

Fig. 8 Excerpt clustered profiles of a project and MBTI suggested roles

This team was tested against another free group (Pleoticus) of the same knowledge level in developing a relatively complex agile project. The results of Artemisas in both technical and process compliance were not very functionality superior, but they showed a noticeably superior overall quality. So, in this context, *AgileTalk* intends to advance in the automation of these results, building a platform for HR experts and project managers to select candidates for ongoing projects, simulate behaviors, or train new apprentices.

4.1 Automating Profiling

.A *TeamBot* profile is obtained through interviews with the apprentice by simulating a typical hiring interview in a software company and refining it through Bayesian inference on indicators raised in agile ceremonies.

The apprentice must have an interview with a specialized HR *TeamBot*, chatting (through the voice interface) about issues related to situations assimilated with the ILS, MBTI, and B5 questionnaires (this conversation is currently limited; however, it can easily be extended with a reasonable effort).

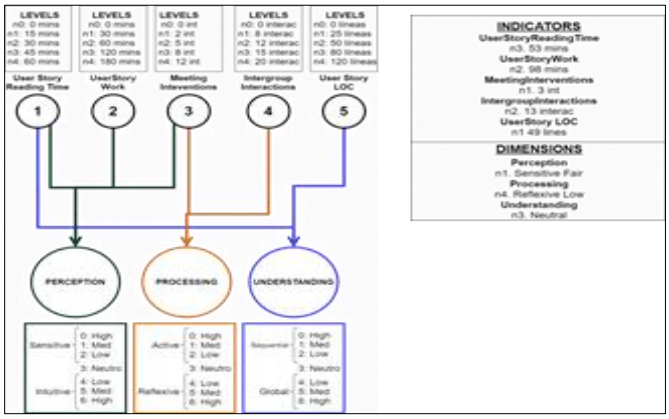

Fig. 9 Partial Bayesian Network and ILS Profile resulting from observing the behavior and performance of a TeamMember

The ILS profile is further refined following the approach that was already described in [8]. This approach is based on Bayesian Networks (BN) to infer, automatically, an apprentice's Learning Style Profile. Fig. 9 presents a partial BN incrementally build showing the indicators used to infer levels in each dimension.

This feature is central in training activities. In the example, if the *"I do not understand"* intention is detected by RASA several times, the assistant *TeamBot* will look for alternative ways for explaining the concept. Simultaneously, given that the apprentice prefers step-by-step explanations, further answers can be switched

[3] The details of these criteria are omitted just for space reasons.

towards a less abstract and more detailed dialog. That is to say that the apprentice tends to have a Sensitive-Sequential profile. If TeamBot had previous knowledge about the Learning Style, then the most suitable *Policy* will be adopted.

Through this mechanism, the HR assistant TeamBot can be configured to follow a custom adaptation policy that adapts to the dialog flow of explanations. For example, the following code snippet describes the custom policy "CustomLearningPolicy" subclass implementing the method *predict_action_probabilities* to provide the appropriate path to follow in the explanation script[4].

```
# Generic policy for selecting type of explanation
class CustomLerningPolicy(Policy):

  # Hook of decision policy dependent on the machine learning technique used
 def predict_action_probability( self,  tracker: DialogueStateTracker,  domain: Domain,
                                 interpreter: NaturalLanguageInterpreter, **kwargs: Any) ) -> PolicyPrediction:
      "Recover the intention in the sequence of the ongoing dialogue
            intent = tracker.latest_message.intent.get (INTENT_NAME_KEY)
            if(not self.answered and intent == 'explain more' )
                  # Criteria for deciding to switch the conversation flow to a sensistive-sequetial profile
                  result = confidence_scores_for (intent, domain)
                  if (result=1)
                        current_dialog_flow = "very high sensitive-sequential"
                  if (result > 0.7)
                        current_dialog_flow = "medium high sensitive-sequential"
                  ....
                  self.answered = TRUE
```

Operationally, the "current_dialog_flow" variable is indeed set to a specific Iterator on the general explanation structure. Furthermore, this feature makes it possible to model subtle but crucial details that help establish comfortable dialogs, such as the involved team members' affinities and personality traits.

4.1.1 Emulating Personality Traits

The strong emphasis on teamwork and self-organizing groups are the main pillars where agile development lies. Therefore, an ambitious goal of *AgileTalk* is to provide a flexible platform enabling exploration of team behavior and performance to the more considerable possible extent. Hence, the ability to emulate team members' behavior considering personality traits and soft skills appears intriguing and potential.

Having the information about the development of a project would make it possible to produce a movie inspired in a living way by how the actors of the process behave during the whole or parts of ceremonies, tasks, or projects. In addition, this information could be valuable for leaders and managers to grasp some problems that would have arisen during the development, assess the velocity of groups or delay causes, or whatever conclusion it potentially could be imagined.

Furthermore, the capability to save textual records of team members' oral interactions provided by the voice interface enables automatic sentiment analysis, expanding the possibility of help to adjust personality profiles [13].

Fig 10 presents a "funny" example of the first steps to implement this support in *AgileTalk*. It shows a centennial *TeamBot*, Matheu, modeled with the following Big Five profile:

< **Extraversion** high (highlighted by the Dynamism sub-dimension),
Affability lofty (huge Cordiality),

[4] This method have been simplified for speace reasons and to simplfy the explanation of the underlying idea

Consciousness low (little Scrupulosity but good Perseverance),
Neuroticism slightly low,
Openness average>[5].

The *Scrum Master TeamBot*, on its part, is modeled by the following profile:

< **Extraversion** high (highlighted by the Dominance sub-dimension),
Affability very low (little Cordiality and Cooperation),
Consciousness high (Scrupulosity and Perseverance over average),
Neuroticism medium-high,
Openness average>

These values are mapped to Avatar's visual patterns (somehow extravagant), taking advantage of Unity power, making it relatively light to build different reactions that can be combined according to the weight of each

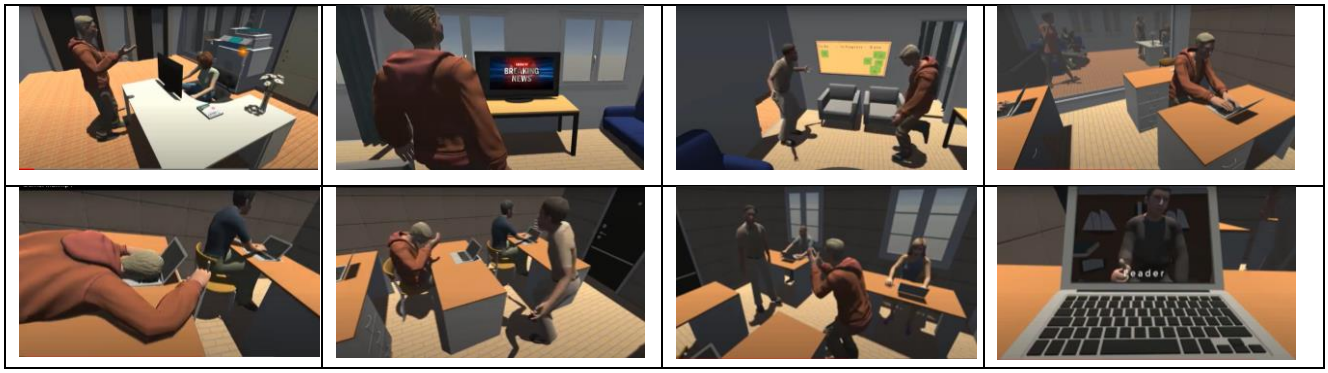

Fig. 10 Excerpts of the Trailer of "A Centennial day in #MicroDesign Soft" (*opera prima)*

test dimension.

The snapshots show different situations emulating a friendly and distracted Matheu behavior and a very responsible and relatively rude annoyed Scrum Master scolding him because of the recurring delays and work style. Matheu passes from oblivious to distraught mood until a happy end.

Indeed, it is necessary to recognize that this is a delicate aspect that depends on many factors and can lead to many different arguments. Notwithstanding, the availability of a little-explored feature like this represents a significant added value from a training platform point of view, opening intriguing research opportunities and the construction of truly influential serious games.

5. Process Monitoring & Control: The Process Action Team

Although it can seem contradictory with the usual interpretation of Agile against bureaucratic approaches like CMMI, successful Agile methods are based on the deployment of strict control mechanisms over its strong emphasis on iterative processes. In this aspect, CMMI process areas like PMC are part of the core enabling practices for successful Agile implementation. The institutionalization of a Process Action Team is an alternative to bring the benefits of process improvement frameworks [4]. The Scaled Agile Framework (SAFE) took this way by reinforcing the Lean philosophy and Feature-Driven Development (FDD) as the organizational framework.

Following this approach, specialized *TeamBots*, the *PATBots*, come into the scene. *PATBots* have associated "synthetic" roles, as managing staff members do have for monitoring and processing process-related events raised by any activity in *AgileTalk* projects. *AgileTalk*'s design is based on an event-driven architecture style as the primary coordination mechanism. A PATBot subscribes to process-specific activity events that trigger the sequence shown in Fig. 11.

[5] The B5 scale ranks each dimension on a scale of 35 to 120 (60 per sub-dimension). In 278 actual samples collected, the average oscillates between 70 and 75 for each dimension of the test.

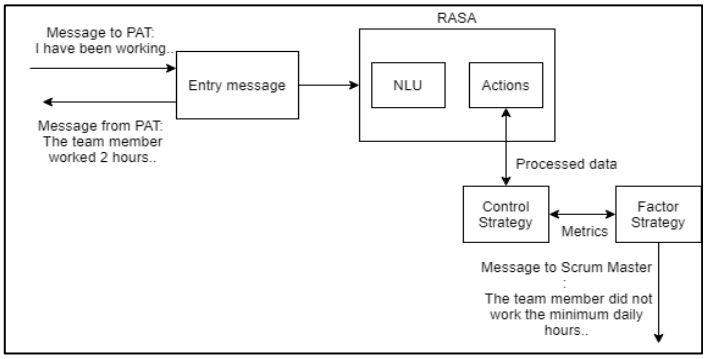

Fig. 11 Process Control Actions Scheme

This activity diagram corresponds to the interchange in natural language among the participant *TeamBots* of the daily meeting described above. This conversation will generate events in JSON format: {'message': 'What have you been working on, Fede?', 'from': 'Rama', 'to': 'Fede" }, for example.

The PATBot's callback interacts with the RASA-side component that determines the following actions to be taken according to the intention detected. For example, the snippet in Fig. 13 shows a possible training set and the stories implementing the related actions

```
def execute_strategies(self, intent_name: str, data: dict) -> str:
 #executes the strategy for the requested intention performing
 #the metric calculation through the factorStrategy component
      result = ""
      if intent_name in self._d_intentStrateg:
         strategies =
self._d_intentStrateg[intent_name]["strategies"]
         for key_strategy, data_strategy in strategies.items():
            result = result + "MÃ©trica calculada: " +
str(key_strategy) + " "
            strategy = data_strategy["Constructor"]
            strategy.process_event(data)
            result = result + " Result: " +
self.process_result(strategy,strategy.calculate_value())
            #Result: The function has the metric calculation in the
form of a String
      return result
```

```
def process_intent(self, kwargs:dict) -> str:
 #Receives the intention along with the data
 #and executes the assigned strategy if it exists
      intent_name = ""
      data = ""
      result = ""
      if "intent" in kwargs:
         intent_name = kwargs["intent"]
         if "data" in kwargs:
            data = kwargs["data"]
            result =
self.execute_strategies(str(intent_name), data)
.....
      return result
```

Fig. 12 Code snippet of ControlStrategy

The first story responds by ignoring the message. The second one, "tasks", triggers the custom action "analize_tasks", which asks the Tracker for the last message sent containing "I have been working on the tasks [*tasks numbers*]" and calls the "*controlStrategy* "python component that will perform the corresponding control. The *ControlStrategy* component will be the ones who process the information and execute the strategy for

```
-intent: what_have_you_been_working_on
  examples: |
    - What have you been working on, 1?
    - What have you been working on, 2?
    …
- intent: working_on_the_tasks
  examples: |
    - I have been working on tha tasks [1:1:2, 4:1:2, 5:1:2]     - I have
been working on tha tasks [2:1:2, 6:1:2, 5:1:2]
…
```

```
- story: ask_working_on_tha_tasks
  steps:
intent:what_have_you_been_working_on
  - action: utter_irrelevant

- story: tasks
  steps:
  - intent: working_on_tha_tasks
  - action: analize_tasks
```

Fig. 13 Partial RASA definition of daily meeting monitoring stories

calculating metrics using Python's reflection capabilities. The "*execute_strategies"* method selects these strategies dynamically through the name of the detected intention. Finally, the "*process_intent*" method executes the corresponding algorithm that computes the related metric (Fig. 12). The described process results are shown in Fig. 14 as an excerpt of the dialog generated from the RASA shell to allow the visualization for debugging.

Go to meeting 1, Rama Message from: Server ProcessActionBot: The agilebot: Rama must go to the meeting: 1 …. ##### Dialog capture >The meeting 1 started: 2021-06-13 01:27:05 >Message from: Server >ProcessActionBot: The start date of the meeting is: 2021-06-13, and the time is: 01:27:05 >Facilitate daily meeting Rama >Message from: Server >ProcessActionBot: The agilebot: Rama will be the ScrumMaster/facilitator of the Daily >What have you been working on, Tincho? >Message from: Rama >ProcessActionBot: Irrelevant for the process >I have been working on the tasks [1:1:3, 2:1:5] >Message from: Fede >ProcessActionBot: Metric taken: ControlTask Result: Task 1 needs 2 more hours/s to be completed. Task 2 needs 4 more hours/s to be completed. The team member worked 2 hours a day.	>The meeting 1 ended: 2021-06-13 01:27:09 >Message from: Server >ProcessActionBot: La fecha de inicio de la reunion es: 2021-06-13, su hora: 01:27:05. La fecha de finalización de la reunion es: 2021-06-13 y la hora es: 01:27:09 >ProcessActionBot: Métrica calculada: EstimatedDeadline Result: La reunion finalizo 896 segundos antes del plazo. >The participation of the team members in the meeting are: [Fede:2, Rama:3,Tincho:2] >Message from: Server >ProcessActionBot: Computed Metric: MeetingParticipations Result: Team member Fede participated 2 times in the meeting. Team member Rama participated 3 times in the meeting. Team member Tincho participated 2 times in the meeting.
Fig. 14	

In closing, this brief example serves as a final tight description of the main ideas underlying *AgileTalk* and the use of the RASA framework.

6. Conclusions and Future Work

This article presents the essentials and a brief description of the first results of a long lasted project pursuing the constant improvement for training future software engineers. The final goal is to set the foundations of a full-fledged platform for building a customized intelligent process-centered development environment. If *AgileTalk* becomes operational in real-world projects, it could enable a more effective tool for development, training, and even hiring software engineers by trying them into simulated projects.

The main contribution presented here is to show that a platform for chatbot development like RASA can become the engine that drives all the behavior of a multi-agent system constituted by *AgileBots*. These are a novel contribution that enables the building of mixed-reality training environments allowing results, in our opinion, much further than the (usually fixed) games proposed in the literature in all its variants.

The personality-based approach to software development has been little explored, mainly in its application as the basis for training and development environments. Undoubtedly, it represents a complex and arguable matter worth to be explored in deep because it is crucial in agile processes.

The current results in this line need to be validated and improved through many controlled experiments. In the context described above, they will be carried out in the following years to reach an industrial prototype that can be experimented with in real-world projects. Particularly, the aid of HR recruiters is currently being explored to define meaningful experimental situations to be modeled and tested.

In conclusion, *AgileTalk*'s objective is undoubtedly ambitious but challenging and intriguing, mainly whether the first results obtained are encouraging. That is, it contains all the ingredients of an exciting research project with reasonable expectations of success beyond the improvement of student's formation we note year after year.

7. Acknowledgments

We thank all the students engaged in the experiment for their enthusiasm, confidence and, fundamentally, make teaching a fascinating and enjoyable experience. Especial thanks to Ezequiel Herrera da Rosa for his commitment to the development of the animations with Unity.

8. References

[1] Arruda D., Marinho M, Souza E, et al., A Chatbot for Goal-Oriented Requirements Modeling, ICCSA 2019, LNCS 11622, pp. 506–519, 2019. https://doi.org/10.1007/978-3-030-24305-0_38

[2] C. Matthies, F. Dobrigkeit and G. Hesse, "An Additional Set of (Automated) Eyes: Chatbots for Agile Retrospectives," 2019 IEEE/ACM 1st International Workshop on Bots in Software Engineering (BotSE), Montreal, QC, Canada, 2019, pp. 34-37, doi: 10.1109/BotSE.2019.00017.

[3] Caprara, G. V., Barbaranelli, C., Borgogni, L., & Perugini, M. (1993). The “Big Five Questionnaire”: A new questionnaire to assess the five factor model. Personality and Individual Differences, 15(3), 281–288.

[4] Dalton, et al., The Agile Profile, CMMI Institute, 2016

[5] Diaz Pace A., Campo M., Exploring Alternative Software Architecture Designs: A Planning Perspective, October 2008, Intelligent Systems, IEEE 23(5):66-77 DOI: 10.1109/MIS.2008.78

[6] Faheem Ahmed, Luiz Fernando Capretz, Salah Bouktif, Piers Campbell. Soft Skills and Software Development: A Reflection from Software. Industry International Journal of Information Processing and Management (IJIPM), Volume4, Number3, May 2013. doi:10.4156/ijipm.vol4.issue3.17

[7] Felder, R. M., & Silverman, L. K. (1988). Learning and teaching styles in engineering education. Engineering Education, 78(7), 674–681.

[8] García, P., Amandi, A., Schiaffino, S., & Campo, M. (2007). Evaluating Bayesian networks' precision for detecting students' learning styles. Computers & Education, 49(3), 794–808.

[9] Goldberg. Language and individual differences: The search for universals in personality lexicons. Review of personality and social psychology, 1981. 2(1), 141-165.

[10] Guillermo Rodríguez · Pablo C. González-Caino · Santiago Resett. Serious games for teaching agile methods: A review of multivocal literature. Computer Applications in Engineering Education May 2021.

[11] Hunt J. Feature-Driven Development. In: Agile Software Construction. 2006 Springer, London. https://doi.org/10.1007/1-84628-262-4_9

[12] Kiku Jones, Lori N. K. Leonard & Guido Lang (2018) Desired Skills for Entry Level IS Positions: Identification and Assessment, Journal of Computer Information Systems, 58:3,214-220, DOI: 10.1080/08874417.2016.1229144

[13] Lescano G., Torres-Jimenez J., Costaguta R., Amandi A., Lara C. Detecting conflicts in collaborative learning through the valence change of atomic interactions. Expert Systems with Applications. In Press, 2021

[14] Myers, Isabel Briggs with Peter B. Myers (1980) Gifts Differing: Understanding Personality Type. Davies-Black Publishing, ISBN=0-89106-074-X.

[15] Narasimhaiah Gorla and Yan Wah Lam, Who Should Work with Whom? Building Effective Software Project Teams, COMMUNICATIONS OF THE ACM June 2004 / Vol. 47, No. 6.

[16] Norman. (1963). Toward an adequate taxonomy of personality attributes: Replicated factor structure in peer nomination personality ratings. *The Journal of Abnormal and Social Psychology*, *66*(6), 574.

[17] Rakesh Kumar Sharma, Manoj Joshi, An Analytical Study and Review of open Source Chatbot framework, RASA, International Journal of Engineering Research & Technology (IJERT) ISSN: 2278-0181 http://www.ijert.org Vol. 9 Issue 06, June-2020

[18] Rodriguez G, Soria Á, and Campo M., (2013). Virtual Scrum: A teaching aid to introduce undergraduate software engineering students to Scrum. Applications in Engineering Education. Wiley Online Library.

[19] S. R. Hedberg, "Intelligent agents: The first harvest of softbots looks promising," IEEE Intelligent Systems, no. 4, pp. 6–9, 1995.

[20] Samer Muthana Sarsam, Hosam Al-Samarraie. A first look at personality in the design of mobile UI for increased satisfaction, SAGE Open, Volume: 8 Issue: 2, April 1, 2018

[21] Scott E., Rodríguez G., Soria Á., and Campo M., (2016), Towards better Scrum learning using learning styles. Journal of Systems and Software, ISSN: 0164-1212, vol. 111, pp. 242-253, Elsevier Science,

[22] Suta, Prissadang & Mongkolnam, Pornchai & Chan, Jonathan & Lan, Xi & Wu, Biting. (2020). An Overview of Machine Learning in Chatbots. International Journal of Mechanical Engineering and Robotics Research. 9. 502-510. 10.18178/ijmerr.9.4.502-510.

[23] Teague Joy, Personality type, career preference and implications for computer science recruitment and teaching. ACSE '98: Proceedings of the 3rd Australasian conference on Computer science education. July 1998 Pages 155–163https://doi.org/10.1145/289393.289416