Towards an ontology-based recommender system for agile practices adoption

M. Luciana Roldán[1], Verónica A. Bollati[2], Silvio M. Gonnet[1]

[1]Instituto de Desarrollo y Diseño (INGAR) – CONICET / UTN
Avellaneda 3657 – Santa Fe – Argentina

[2]UTN Fac. Reg. Resistencia / CONICET,
Resistencia, Argentina.

{lroldan,sgonnet}@santafe-conicet.gov.ar, vbollati@gmail.com

Abstract. *Choosing the right agile practices to be adopted in a software development process is complex, especially for organizations with little experience and a high turnover of human resources. To select the best practices, it is necessary to consider the characteristics that describe the team, the project, and the context in which they will be applied. In this paper, we propose an ontology-based agile practices recommender. The ontology was built and populated with knowledge obtained from software projects of Argentine organizations. The recommender applies the FP-Growth algorithm to find the most suitable practices for the target profile. In addition, it discovers potential conflicts between the profile and the recommended practices.*

1. Introduction

In the last decades, the software development industry has been increasingly adopting agile methods and practices instead of traditional software development methods as they are more flexible and can bring benefits such as handling requirements changes, productivity gains, and business alignment [Campanelli et al. 2018]. In fact, the well-known survey of Digital.ai [Digital.ai 2021] reports that the two most urgent reasons for adopting Agile are the speed and flexibility required by working environments that continue to be both unpredictable and volatile (64% Enhance ability to manage changing priorities; 64% Accelerate software delivery). These are closely followed by a continued need to focus on alignment across teams to streamline the software delivery process (47% Increase team productivity).

Digital.ai survey also shows that organizations still face a variety of challenges as they introduce agile techniques, practices, and tools. In fact, 30% of respondents identified no fewer than ten different challenges faced while adopting Agile. The key challenges organizations face when adopting Agile have remained largely unchanged for the past several years. Challenges with organizational culture, resistance to change, and lack of skills and experience continue to be problems.

Choosing the right strategy for the adoption of agile practices is key for organizations to take advantage of the benefits of agile and overcome the common problems found in the adoption process and achieve their business goals. Currently, the main difficulty in the agile adoption process lies in selecting the right set of practices for each situation. Numerous research efforts have been reported to address this problem, but there is still no definitive answer [Kurapati et al. 2012; Letelier and Penades 2017; Nishijima and Dos Santos 2013].

The complexity of the agile adoption process is high, especially for organizations with little experience and a high turnover of human resources. Often, companies hire external coaches to provide initial training and kick-start agile development [Stray et al. 2020]. However, consulting costs are high, especially for organizations in countries with economic difficulties, such as Argentina. To select the right practices for an organization, it is important to have experience with agile development projects to know which practices will give the best results, and to consider the characteristics that describe the team, the project and the context in which the practices will be applied. In today's organizations, it is common for two teams in the same organization to have different characteristics in terms of the background of their members, the history of projects developed in the past, and the environment in which they are immersed (such as legal, budgetary, technological), which adds complexity when selecting the optimal set of practices for a particular "profile" (team/project/context).

To overcome those issues, in this work a preliminary agile practices recommender system is proposed. The system is based on an ontology that represents agile knowledge gained from literature, technical websites and forums, and industry experiences. It constitutes a decision support tool for the process of adopting agile practices. The system was developed on top of an Agile Practices Ontology (AgiPrOnt) to provide an organization or team with the N agile practices best suited to adopt in a particular software development project and context. The contributions of this paper are: an ontology of agile practices, a recommender system based in that ontology, and a conflict resolution mechanism.

The rest of the paper is organized as follows. Section 2 presents related works regarding ontology-based recommending techniques and approaches for Agile process adoption. In Section 3, we present the research methodology followed. In Section 4, we present an overview of the Agile Practices Recommender system. Then, the details of construction and population of the AgiPrOnt ontology are given, and we expose the recommendation algorithm and the conflicts detection approach. In Section 5, some aspects of the proposal are evaluated, and the limitations are discussed. Finally, in Section 6 the conclusions are presented.

2. Related works

Recommender systems (RS) have been proposed as a decision-making strategy for users in complex information environments. Recommender systems [Isinkaye et al. 2015] handle the problem of information overload that users normally encounter by providing them with personalized, exclusive content and services. They are also a means of assisting and augmenting the social process of using recommendations of others to make choices when there is not sufficient personal knowledge or experience.

Ontologies play an important role in knowledge representation as well as in knowledge sharing and reuse in recommendation systems. Previous studies have shown that the incorporation of an ontology domain knowledge to a recommender improves the accuracy and quality of recommendations as well as alleviates other drawbacks associated with conventional recommendation techniques such as cold-start and rating sparsity problems. The effectiveness of ontology-based recommender systems depends mainly on the completeness and accuracy of knowledge maintained in the ontology domain knowledge [Tarus et al. 2018]. RS use domain ontologies to establish the relationships between users and their preferences about the recommendation subject. The semantic information of an item includes the attributes, the relationships among the items, and the relationship between meta-information and items. Many RS employ domain ontologies to measure user preferences with content elements. Moreover, using knowledge about items and users helps to produce a recommendation based on knowledge and reasoning about which item meets the needs of the users [Burke 2000].

There has been prolific literature regarding approaches for agile process implementation and tailoring. A probabilistic model to identify problems related to a project and the team is defined in [Perkusich et al. 2013]. The approach is aimed to help Scrum Masters to be aware of the project's problems and have enough information to guide the team and improve the project's chances of success. The model is a Bayesian network that represents a Scrum-based software development project composed by one development team. One of the limitations of this work is that it focuses on the Scrum framework, which makes it useful only for development projects that apply Scrum.

AgileRoadmap [Letelier and Penades 2017] is an approach for implementing ASD in companies. The proposal includes a model and a strategy for agile implementation, which integrates the most popular agile practices, focusing on the practice application and not on the application of a particular method.

In the literature, some authors propose ontologies, taxonomies, or metamodels to describe ASD knowledge. However, most of these works focus on one or two methods or a bunch of practices and do not cover comprehensively the knowledge related to ASD, regarding the variety of agile practices, characteristics of the teams that use them, and the projects and the environments where they are applied [Júnior et al. 2021]. [Minoli 2011] proposes a framework that suggests solutions tailored to the configurations of projects in the context of A-GSD (Global Agile Software Development). The ontology-based solution framework aims to classify, organize, and unify all the existing knowledge on A-GSD, for which it includes an ontology that represents the knowledge in terms of agile and distributed projects features, and a tool that uses the knowledge of this ontology to make available to the community the existing knowledge about A-GSD. [Ortega-Ordoñez et al. 2019] suggested a common and consistent terminology that allows sharing the knowledge generated from implementing agile approaches in the software processes. The proposal focuses on facilitating the assessment of the agility of the software processes, from the identification of the relationships between the elements of the software processes and the agile principles and values. These ontologies lack concepts to characterize the teams, projects, and contexts where agile processes were performed making it difficult to reuse the shared knowledge in the ontology.

[Kiv et al. 2019] proposed an ontology for representing agile methods knowledge, so that the knowledge and experience on agile adoption reported in the literature may be reusable and systematic. This work has similar goals as ours, as it includes concepts to define certain characteristics of the situation of an organization, its goals, and the practices the organizations want to adopt. The approach is based on the knowledge of ASD cases studies and a set of inferences rules. It provides a list of potential problems that could arise when a team adopts a practice, and, for each problem, propose a solution. However, it requires the user to have expert knowledge about practices to adopt, since he/she must decide in advance which practices the team will adopt. In addition, the tool does not provide support to find teams/organizations with similar characteristics to the user organization.

To the best of our knowledge, no proposal integrates a recommending algorithm based on team, project, and context characteristics, with ontologies to suggest a starter pack of practices to a team initiating a new agile project.

3. Methodology

We employ the Design Science Research paradigm, as the research method adopted in this work, which is concerned with extending "human and organizational capabilities by creating new and innovative artifacts" [Peffers et al. 2007]. It comprises five steps, which are organized in an iterative process: (i) Problem identification and Motivation, (ii) Definition of the objectives for a solution, (iii) Design and Development, (iv) Demonstration, (v) Evaluation, and (vi) Communication.

Regarding the *Problem identification and motivation* step, the problem was identified by the second author when working as a consultant in some organizations on the Northeast region of Argentina. Additionally, the "lack of skills and experience" has been identified as a barrier to Agile adoption from multiple studies [Digital.ai 2021; PMIBA 2020]. The problem means that organizations are forced to invest heavily in training programs and hiring coaches to gain insight into which practices to select when facing a new project.

At the *Definition of the objectives for a solution* step, we decided to develop *AgiPront*, an ontology-based recommender system for agile practices adoption. As requirements of the proposed approach, we defined that it must: (R1) allow identifying the team's characteristics, the project characteristics, and the context in which the new project will be inserted, (R2) provide meaningful and comprehensive data on agile practices, as well as an assessment of the impact of their use on past completed projects, and (R3) recommend a set of agile practices appropriate to the characteristics of the team, project, and context of the organization that uses the tool.

4. The Agile Practices Recommender System

In this section, we address the *Design and Development* step prescribed by the followed research methodology, outlining the system architecture first. The main stakeholders of the Agile Recommender Systems are: i) the *Target Agile Profile* (*TAP*), a person, team or organization that requires support to select the set of best practices to adopt for a given project and context; ii) the *Expert*, who is the responsible to feed the ontology with ASD knowledge from several sources like the literature, web forums, self-experience, open projects repositories, etc. Another actor (not shown in Figure 1) is the *Surveyed Agile Profile*, which interacts with the system by providing data on cases studies, and data from completed ASD projects.

In Fig. 1 the central component of the RS is the semantic model, formalized in an ontology, which represents the concepts involved in Agile Software Development. A web questionnaire is used to collect part of the data required to feed the ontology, by inquiring to diverse agile organizations about their characteristics, the practices they employed in their projects, and the contribution of each practice adopted to the success of the project. The Profile Loader component populates the ontology from the knowledge obtained with the questionnaire. A SPARQL endpoint is a service that allows applications to perform SPARQL queries on data stored in RDF format. Through the Agile Knowledge Interface, users can access the ontology and employ it for learning/training purposes. The Recommender Algorithm consists of three parts: a) a profile matching stage, where the characteristics of the Target Agile Profile are identified and used to find other similar profiles in the knowledge base; b) a stage in which a set of the most successful practices used jointly by similar profiles is obtained, , which constitute the practices that are candidates for recommendation; and c) a conflicts detection and ranking stage, where rules are applied to discover potential conflicts among the candidate practices.

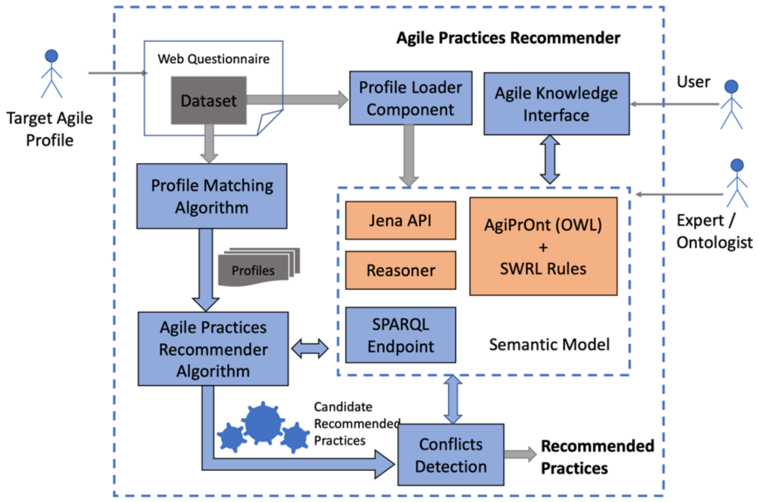


Figure 1. Overview of the Agile Practices Recommender System

4.1. The AgiPrOnt ontology

To develop AgiPrOnt, an ad-hoc methodology was applied that comprises four stages, which are common to most popular ontology development methodologies [Suárez-Figueroa et al., 2012]: i) an Ontology requirements specification stage, which consists of identifying the purpose for constructing the ontology and set its scope; ii) a Conceptualization stage, which consists of organizing the involved concepts in an informally perceived view of the domain and, then, converting it in a semi-formal UML-based specification; iii) a Formalization stage, which comprises the implementation of the ontology using a formal language, as well as populating it with knowledge from different sources; and iv) an Evaluation stage, which allows making a technical judgment of the ontology quality and its usefulness with respect to the requirements specification, competency questions and/or the real world.

The requirements specification stage began with the elicitation of a set of competency questions, which, due to lack of space, are given in [Roldan et al. 2022] (*AgiPrOnt Competency questions* file).

At the conceptualization stage, we decided to build AgiPrOnt as a domain-specific ontology based on the concepts defined by the Software Process Ontology

(SPO) [Oliveira Bringuente et al. 2011]. SPO is a core ontology that is part of SEON (Software Engineering Ontology Network) and provides the general concepts for software processes. SEON [Borges Ruy et al. 2016] provides a network of Software Engineering reference ontologies, and mechanisms to derive and incorporate new integrated subdomain ontologies into the network. The foundational layer of SEON, is the Unified Foundational Ontology (UFO) [Guizzardi 2005], which is developed based on a number of theories from Formal Ontology, Philosophical Logics, Philosophy of Language, Linguistics and Cognitive Psychology. Then, the definition of the concepts and relationships of AgiPrOnt was made by extending the specific concepts and relationships included in SEON network ontology, particularly in UFO and SPO.

From the elicited competency questions, we identified the terms of the ASD domain. Initially, we generated a list of terms without considering the existence of any overlapping, the possible relationships among them, or the properties that characterize each concept. Then, each term was defined as an extension of the general concepts in the upper ontologies.

To organize the concepts in *AgiPrOnt*, four semantic submodels were defined (Fig. 2): the Fundamental Agile Concepts Model (*FACM*), the Agile Practices Model (*APrM*), the Agile Characteristics Model (*AChM*), and the Agile Assessment Model (*AAM*). The *FACM* Model comprises the definitions of fundamental concepts of ASD and a canonical model for representing agile developing processes. *Agile Values* are part of the *Agile Manifesto*, which defines the philosophy of agile [Beck. K. et al., 2001]. From these values, twelve *AgilePrinciples* are derived, which enable mapping the agile practices to the business goals the organizations want to achieve. Agile values and principles are implemented through *Agile Practices*. An agile practice is an activity or a mode of working, which can be applied repeatedly, thus performed regularly (in the case of an activity) or enforced systematically (in the case of a mode of working) [Meyer 2014].

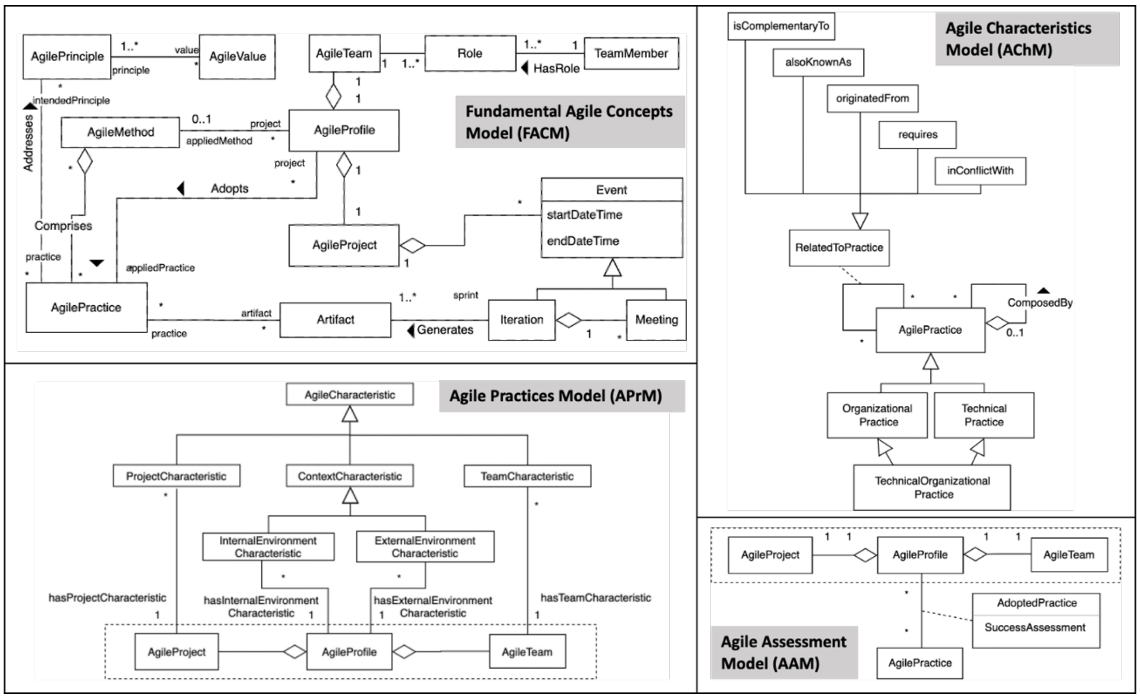


Figure 2. Ontology Conceptual Model

The application of agile practices gives rise to different artifacts or work products (*Artifact*). Some practices can be part of a specific *Agile method*, although it is not required for a practice to be part of a method (*Comprises* relationship).

The concept of *AgileProfile* is key in this proposal, as it acts as the "container" of the concepts *AgileTeam* and *AgileProject*. An agile project is a software development project that follows an extremely iterative and incremental life cycle, where the development team works in a highly collaborative and self-organized manner. An agile project is composed by events. Events are time-boxed, which means that they have a fixed and predefined maximum duration and can be classified as *Iterations* or *Meetings*. An *Agile Team* is defined as a set of persons involved in the development of the project. An organization may have multiple projects in progress, and different teams in charge of their development. Each *Team Member*, who participates in an agile team, has a specific *Role* (a set of functions that are fulfilled within the agile team).

The *APrM* model represents that an *AgilePractice* can be an *OrganizationalPractice*, a *TechnicalPractice,* or both (*TechnicalOrganizationalPractice*). While organizational practices affect the cost, schedule and team sustainability of a project, the technical practices support product quality and effectiveness. The model includes some relationships among practices, which are defined by the Agile Alliance Glossary [Agile Alliance [S.d.]]: *composedBy alsoKnownAs*, and *originatedFrom*. Other relationships included in the model are: *inConflictWith* (to indicate that two practices cannot coexist in the same project), *isComplementaryTo* (to represent that if a practice A is adopted, then, it is highly recommended to adopt also practice B) and *requires* (to represent that if practice A is adopted, practice B must be also adopted for A to work properly).

The *AChM* Model includes the concepts for representing the properties or attributes that characterize to the agile teams, the projects they carry on, and their contexts. The central concept is *AgileCharacteristic*, which is specialized in *TeamCharacteristic* (how the team is composed and how their members interact), *ProjectCharacteristic* (budget, schedule, resources, and technological aspects of the system or software to be developed), and *ContextCharacteristic*. The last one refers to the environment of a Team+Project pair, and it is specialized in *ExternalEnvironmentCharacteristic* (aspects from the outside of the organization) and *InternalEnvironmentCharacteristic* (aspects from the inside of the organization).

The *AAA* model represents the knowledge about the success of the practices adopted by agile profiles. The recorded knowledge from past experiences is useful for new agile teams that face a new agile project in a new context. To explicitly represent this knowledge, the model incorporates a relationship between the *AgileProfile* and *AgilePractice* concepts. This relationship, called *AdoptedPractice*, also maintains information about in what extent a given practice contributed to the overall success of a project (*SuccessAssesment* attribute).

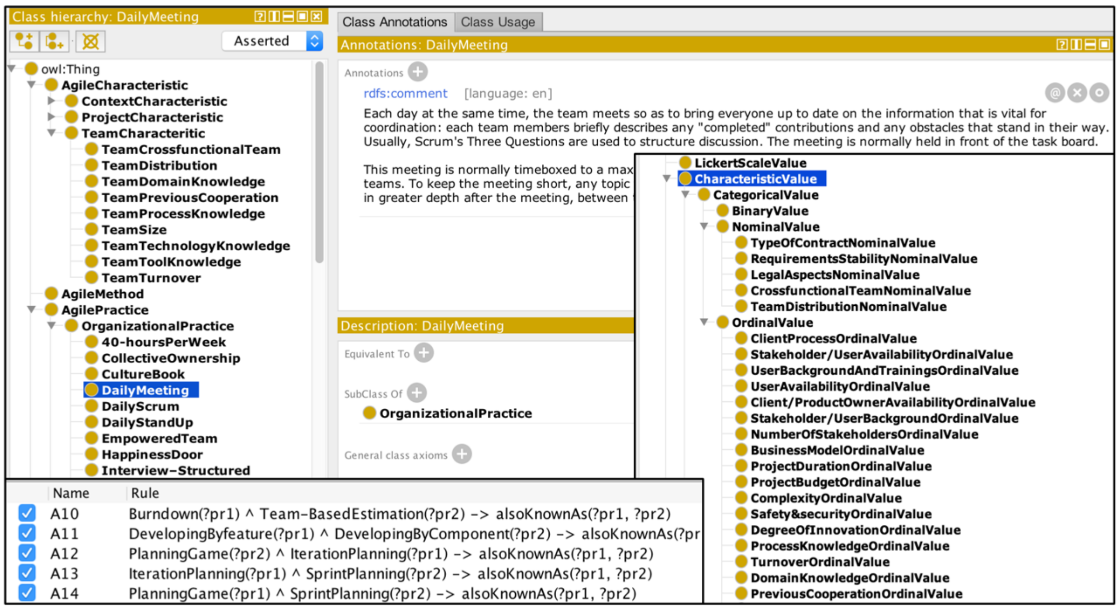
The next step was the formalization of the conceptual model in OWL, in terms of classes, individuals, and properties (Fig. 3), using Protegé[1]. When implementing the *APrM* model in OWL, two possible design decisions were considered on how to define

[1] https://protege.stanford.edu/

the agile practices. The first option was to create an instance of *AgilePractice* class for each existent practice. The alternative option was to create a subclass by each existent agile practice extending the *AgilePractice* concept, and then generate a singleton instance for each subclass. We selected the second option because it allows us to define richer rules in the ontology and work with a reasoner to get meaningful results from the inferences. Over 70 agile practices were collected and consolidated from various sources (literature, websites, and agile community forums), and defined in AgiPrOnt [Agile Alliance [S.d.]; Campanelli et al. 2018; Digital.ai 2021]. As there is no consensus in the literature on the names of agile practices, some relationships of "aliases" were established by means of inference rules. In addition, annotation properties were added to the practices (*definition*, *benefits*, and *pitfalls*).

Like practices, profiles' characteristics were also defined as classes. The complete set of practices and characteristics that were regarded in this proposal are available in the *Datasets* folder in [Roldan et al. 2022]. To represent that an agile profile has certain characteristics, the *AgileCharacteristic* instances are related to an instance of *AgileProfile* through the *hasCharacteristic* object property (either *hasTeamCharacteristic*, *hasProjectCharacteristic*, *hasExternalEnvironmentCharacteristic*, or *hasInternalEnvironmentCharacteristic* object properties). Characteristics have values (*hasValue* object property), which are used to assign a specific meaning to the characteristic for a profile. The values can be of different types (Fig. 3): categorical values (binary, nominal, or ordinal) and numerical values. While nominal values represent discrete units and are used to label variables that have no quantitative value and no order, ordinal values represent discrete and ordered units. On the other hand, numerical values can be continuous, interval, or discrete.

The *AdoptedPractice* relation (Fig. 2) was reified in an OWL class called *AdoptsPracticeRelationship*, in such a way that it is possible to assign it a property value to maintain the assessment value given by a profile to the practice based on the success of the project. The assessment value is represented on a *Likert scale* value.



```
owl:Thing
  AgileCharacteristic
    ContextCharacteristic
    ProjectCharacteristic
    TeamCharacteritic
      TeamCrossfunctionalTeam
      TeamDistribution
      TeamDomainKnowledge
      TeamPreviousCooperation
      TeamProcessKnowledge
      TeamSize
      TeamTechnologyKnowledge
      TeamToolKnowledge
      TeamTurnover
  AgileMethod
  AgilePractice
    OrganizationalPractice
      40-hoursPerWeek
      CollectiveOwnership
      CultureBook
      DailyMeeting
      DailyScrum
      DailyStandUp
      EmpoweredTeam
      HappinessDoor
      Interview-Structured
```

rdfs:comment [language: en]
Each day at the same time, the team meets so as to bring everyone up to date on the information that is vital for coordination: each team members briefly describes any "completed" contributions and any obstacles that stand in their way. Usually, Scrum's Three Questions are used to structure discussion. The meeting is normally held in front of the task board.

This meeting is normally timeboxed to a max
teams. To keep the meeting short, any topic
in greater depth after the meeting, between

SubClass Of: OrganizationalPractice

```
LickertScaleValue
CharacteristicValue
  CategoricalValue
    BinaryValue
    NominalValue
      TypeOfContractNominalValue
      RequirementsStabilityNominalValue
      LegalAspectsNominalValue
      CrossfunctionalTeamNominalValue
      TeamDistributionNominalValue
    OrdinalValue
      ClientProcessOrdinalValue
      Stakeholder/UserAvailabilityOrdinalValue
      UserBackgroundAndTrainingsOrdinalValue
      UserAvailabilityOrdinalValue
      Client/ProductOwnerAvailabilityOrdinalValue
      Stakeholder/UserBackgroundOrdinalValue
      NumberOfStakeholdersOrdinalValue
      BusinessModelOrdinalValue
      ProjectDurationOrdinalValue
      ProjectBudgetOrdinalValue
      ComplexityOrdinalValue
      Safety&securityOrdinalValue
      DegreeOfInnovationOrdinalValue
      ProcessKnowledgeOrdinalValue
      TurnoverOrdinalValue
      DomainKnowledgeOrdinalValue
      PreviousCooperationOrdinalValue
```

	Name	Rule
✓	A10	Burndown(?pr1) ^ Team-BasedEstimation(?pr2) -> alsoKnownAs(?pr1, ?pr2)
✓	A11	DevelopingByfeature(?pr1) ^ DevelopingByComponent(?pr2) -> alsoKnownAs(?pr
✓	A12	PlanningGame(?pr2) ^ IterationPlanning(?pr1) -> alsoKnownAs(?pr1, ?pr2)
✓	A13	IterationPlanning(?pr1) ^ SprintPlanning(?pr2) -> alsoKnownAs(?pr1, ?pr2)
✓	A14	PlanningGame(?pr1) ^ SprintPlanning(?pr2) -> alsoKnownAs(?pr1, ?pr2)

Figure 3. Partial view of the ontology model formalized in OWL

The OWL ontology was enhanced regarding completeness, expressiveness, and logic by adding SWRL (Semantic Web Rules Language) rules. The first set of rules was specified to infer associations between instances, and other ones to express alias relationships (*alsoKnownAs* object property) between practices (Fig. 3).

The next step was to *populate the ontology*. First, an *owl:allDifferents* axiom was added to the ontology to explicitly state that all individuals are different. The sources of knowledge to populate the ontology were diverse. Mainly, several case studies and data were gathered during a previous research project in which one of the authors participate. In that project a framework and a tool were developed to support management and evaluation of ASD projects, with the aim of facilitating the adoption of quality assurance practices [Pinto et al. 2018]. In this project, 47 case studies of participating agile organizations were collected.

A structured web questionnaire helped to systematize the task of characterizing the agile profiles for the different case studies collected. The questionnaire is made up of a set of closed questions, which provide the possible response values for each characteristic of a profile. Additionally, the questionnaire inquiries about the practices that a profile has adopted or implemented during an ASD process and asks the respondent to rate or evaluate the contribution of each practice to the success of the project. One participant from each project answered the questionnaire. In other cases, the questionnaire was answered by a consultant that had been involved with the organization during the project.

The first set of questions aims to characterize the team, the attributes of the developed project (system/software), and the characteristics of the context (internal and external environment) in which the team and project are immersed. The possible answers are categorical, and their values can be, according to the characteristic considered, ordinal, nominal, or binary types. The ordinal and nominal values take integer values between 1 and 5, where each number has an associated meaning. For example, the team characteristic "*Crossfunctional Team*" takes the nominal values "*siloed teams*" (1), "*most members are specialists*" (2), "*50/50*" (3), "*most members have multiple skills*" (4) and "*full crossfunctional*" (5). Each possible value is mapped to instances of *CrossfunctionalTeamNominalValue* (a subclass of *NominalValue*) in the ontology. The project characteristic "*Project budget*" takes ordinal values, such as "< *$1 million*", "*$ 1-2 millions*", "*$ 2-3 millions*", "*$ 3-4 millions*", and "*$ 5 millions Ordinal*". "*Management support*" is an internal context characteristic, that takes the binary values "No" or "Yes". The *Datasets* folder in [Roldan et al. 2022] includes the complete definition of the values of the characteristics. Most of the characteristics and their values were taken from [Campanelli et al. 2018], then adapted and categorized to design the questionnaire. The data gathered with the questionnaire were used to create instances (individuals) of the *AgileProfile* class, and the instances of the correspondent *TeamCharacteristic*, *ProjectCharacteristic*, and *ContextCharacteristic* subclasses.

The second set of questions inquiries about what practices the agile team adopted and employed during the development of the project. For each selected practice, the participant is asked: "Did the practice contribute to the overall success of the project?". The answer provided must be a Likert scale value, which is associated with a numerical score: *Strongly Disagree* (1), *Disagree* (2), *Neutral* (3), *Agree* (4), and *Strongly Agree* (5). These values provide a rating or assessment based on the respondent's perception of the contribution of the practice to the overall success of the project. The data collected with the questionnaire are available in the *Datasets* folder in [Roldan et al. 2022] (*Profiles.xls* and *AdoptedPractices.xls* files). They were used to populate the ontology by employing the *Profile Loader* component (developed in Java and the Apache Jena API).

4.2. The recommending algorithm

The proposed recommendation algorithm has two phases: 1) to find the right set of candidate practices for an agile objective profile, 2) to detect conflicts, discard practices and obtain the final recommendation. To find the set of candidate practices to recommend, the algorithm follows the following steps:

Step 0: creation of the *target* agile profile. To instantiate the target profile the User stakeholder answers the first set of questions in the questionnaire. An instance of the class *TargetAgileProfile* is generated (AP_T) and the instances for their characteristics, which are related through an object property to their assigned values.. This profile has particular values for the Team / Project / Context characteristics $\{TC_{1T}, TC_{2T}, ..., TC_{tT}, PC_{1T}, PC_{2T}, ..., PC_{pT}, ..., CC_{1T}, CC_{2T}, ..., CC_{cT}\}$. The user also is asked about the *minimum rating value* (MRV) the algorithm should employ to consider as successful the adoption of a certain practice by a profile.

Step 1. Find the *k* agile profiles AP_i most similar to the target profile AP_T, calculating the similarity according to the values of their characteristics. The result of this step is a set of the *k* agile profiles $\{AP_1, AP_2, ..., AP_k\}$ most similar to AP_T. Given that the values of the characteristics are categorical (nominal, ordinal and binary values), the similarity algorithm uses a distance function for categorical/binary data (*SimpleMatchingSimilarity*).

Step 2. For each AP_i in $\{AP_1, AP_2, ..., AP_k\}$, retrieve the practices (P_j) each profile adopted during the development of a project and the corresponding assessment about its degree of success. These data are represented by a matrix of ratings, where the rows (i index) are the similar agile profiles, and columns are the practices (*j* index). In other words, each row in the matrix is the tuple of ratings ($r_{i,1}, r_{i,2}, ..., r_{i,j}, ..., r_{i,m}$). A rating $r_{i,j}$ is a value on the Likert scale between 1 and 5 given by a profile *i* to a practice *j*. If an agile practice P_j has not been adopted by the *APi* profile, a rating $r_{i,j}$ with a value of 0 is considered.

Step 3. The $r_{i,j}$ values (ratings) are converted from nominal (Likert scale value or zero) to binary (True/False). That is made using a mapping function, which, in case the value of a $r_{i,j}$ rating is between the selected MRV and 5 (the highest Likert scale value), it becomes true, otherwise false. The result is a new "dataset" that for each profile (rows) and each practice (columns) has "*true*" or "*false*", being *false* if the practice was not adopted or the success rating was low, or *true* in case it was adopted with a neutral or positive degree of success for the project.

Step 4. Calculate the frequently co-occurring successful practices used by the similar agile profiles, using the *FP-Growth algorithm. FP-Growth algorithm* (derived from A-Priori) is an efficient algorithm for calculating frequently co-occurring items in a transaction database. FP-Growth algorithm is commonly applied in domains such as market basket analysis, but also has been applied to software engineering problems

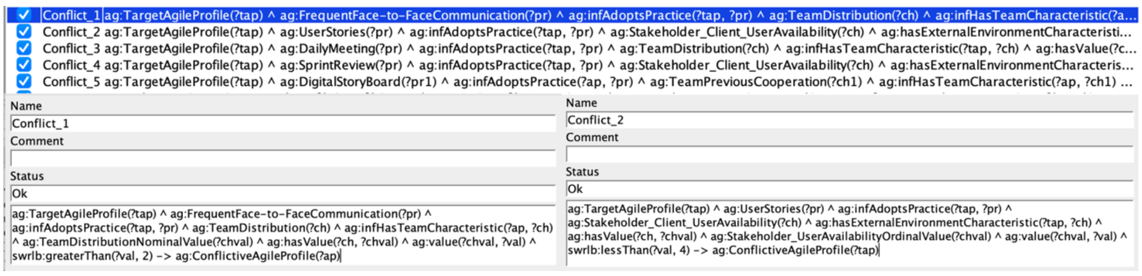
[Muhairat et al. 2020]. To apply FP-Growth algorithm in the ASD domain, we consider the following:

- an item = an agile practice
- a transaction = a complete set of agile practices successfully adopted by an organization/team in the development of a project, in a certain context.
- an itemset = any subset of practices that were applied together by the same agile profile. For example, two possible *itemsets* could be (*daily meeting*, *continuous delivery*, *face-to-face communication*, *poker planning*) and (*daily meeting*, *continuous delivery*, *poker planning*).

The input of the FP-Growth algorithm is the result obtained in step 3, which is in the format "items in dummy coded columns". After applying it, the frequently occurring practices are provided. The *itemset of practices* with the maximum length and an acceptable support value contains the candidate practices CP_{kT} for the target agile profile $\{CP_{1T}, \ldots, CP_{iT}, \ldots, CP_{mT}\}$. The steps 1 to 4 of the algorithm were implemented in RapidMiner[2] using a plain dataset and not directly the instances in AgiPrOnt, for better performance [Roldan et al. 2022].

The set of candidate practices needs to be validated to avoid recommending practices that conflict with each other or with the target profile. The conflict resolution phase is addressed with the usage of OWL DL and SWRL rules. The technique for conflict detection employed is based on introducing inconsistency in the ontology and the use of disjoint classes [Alcaraz Calero et al. 2010]. Since the models are described in OWL DL which is based on Description Logic, a DL reasoner is used to deal with inconsistencies, that is, two conflicting facts are held in the knowledge base. When an inconsistency is detected by the DL reasoner, it fires a conflict detection alert.

To detect a possible conflict, we add to the ontology a class called *ConflictiveAgileProfile*, an "artificial" concept for the domain. The *ConflictiveAgileProfile* class is defined as *disjoint with* the *TargetAgileProfile* class. An OWL disjoint axiom (*disjointWith*) is defined between the two classes, which determines that these classes cannot have instances in common. Therefore, if a disjoint statement is specified for two different classes, a reasoner can deduce an inconsistency when an individual is stated to be an instance of both classes. For the conflict detection mechanism to work, the *Expert* user has to define a set of SWRL rules (Fig. 4), which express potential conflicts between practices, or between practices and profiles that have certain characteristics.



```
✓ Conflict_1 ag:TargetAgileProfile(?tap) ^ ag:FrequentFace-to-FaceCommunication(?pr) ^ ag:infAdoptsPractice(?tap, ?pr) ^ ag:TeamDistribution(?ch) ^ ag:infHasTeamCharacteristic(?a...
✓ Conflict_2 ag:TargetAgileProfile(?tap) ^ ag:UserStories(?pr) ^ ag:infAdoptsPractice(?tap, ?pr) ^ ag:Stakeholder_Client_UserAvailability(?ch) ^ ag:hasExternalEnvironmentCharacteristi...
✓ Conflict_3 ag:TargetAgileProfile(?tap) ^ ag:DailyMeeting(?pr) ^ ag:infAdoptsPractice(?tap, ?pr) ^ ag:TeamDistribution(?ch) ^ ag:infHasTeamCharacteristic(?tap, ?ch) ^ ag:hasValue(?c...
✓ Conflict_4 ag:TargetAgileProfile(?tap) ^ ag:SprintReview(?pr) ^ ag:infAdoptsPractice(?tap, ?pr) ^ ag:Stakeholder_Client_UserAvailability(?ch) ^ ag:hasExternalEnvironmentCharacteris...
✓ Conflict_5 ag:TargetAgileProfile(?tap) ^ ag:DigitalStoryBoard(?pr1) ^ ag:infAdoptsPractice(?ap, ?pr) ^ ag:TeamPreviousCooperation(?ch1) ^ ag:infHasTeamCharacteristic(?ap, ?ch1) ...
```

Name	Conflict_1
Comment	
Status	Ok
	ag:TargetAgileProfile(?tap) ^ ag:FrequentFace-to-FaceCommunication(?pr) ^ ag:infAdoptsPractice(?tap, ?pr) ^ ag:TeamDistribution(?ch) ^ ag:infHasTeamCharacteristic(?ap, ?ch) ^ ag:TeamDistributionNominalValue(?chval) ^ ag:hasValue(?ch, ?chval) ^ ag:value(?chval, ?val) ^ swrlb:greaterThan(?val, 2) -> ag:ConflictiveAgileProfile(?ap)

Name	Conflict_2
Comment	
Status	Ok
	ag:TargetAgileProfile(?tap) ^ ag:UserStories(?pr) ^ ag:infAdoptsPractice(?tap, ?pr) ^ ag:Stakeholder_Client_UserAvailability(?ch) ^ ag:hasExternalEnvironmentCharacteristic(?tap, ?ch) ^ ag:hasValue(?ch, ?chval) ^ ag:Stakeholder_UserAvailabilityOrdinalValue(?chval) ^ ag:value(?chval, ?val) ^ swrlb:lessThan(?val, 4) -> ag:ConflictiveAgileProfile(?tap)

Figure 4. SRWL rules to conflict detection

[2] https://rapidminer.com/

A rule-based reasoner is used to perform inferences on the ontology by applying the SWRL rules. Each conflict rule defined for detecting a conflict between the target profile AP_T and a candidate practice CP_j has an antecedent that is the conjunction of facts that cause a conflict, and a consequent that is the inferred fact *ConflictiveAgileProfile*(AP_T). For each candidate practice CP_{jT} the following steps are performed:

i. An axiom is added to the ontology that sets an *AdoptsPracticeRelation* "relationship" between the target profile AP_T and the singleton instance corresponding to the practice CP_j. In this way, it is "temporarily" stated in the ontology that the AP_T (target agile profile) adopts the practice CP_j.

ii. The reasoner is synchronized, which implies that the rules for detecting conflicts are applied. If the antecedent of a conflict detection rule is true, the *TargetAgileProfile* instance (AP_T) is classified as *ConflictiveAgileProfile*. However, since previously the AP_T had been created as an instance of *TargetAgileProfile*, an inconsistency is reported since the AP_T instance cannot be instance of two disjoint classes. This means that the adoption of the CP_j by the target profile AP_T incurs in a practice-profile conflict, or practice-practice conflict, so the candidate practice CP_j should be dismissed.

iv. The user is informed about the existent conflict, and the practice is eliminated from the recommendation set. When the conflict is between two practices, the user can choose the practice to discard. Then, the axioms added in step *i* are deleted.

Some examples of SWRL rules for conflict detection are below. Due to lack of space, the SRWL rules are detailed in the OWL ontology in [Roldan et al. 2022].

Rule example 1: An agile team that wants to apply the practice "Frequent face-to-face communication" needs all its members to work in the same physical location (city or office), or at most, in the same country (which means that traveling for meetings is feasible). To express that in a rule, the characteristic *TeamDistribution* of the target profile should not have the values greater than 2, being "Same location" or "Same country" categorical/nominal values with values 1 and 2.

Rule example 2: To successfully employ the practice "User Stories", the target profile should have the external context characteristic "Product Owner/Client Availability" with value "Medium-to-High" or "High" (the categorical/nominal values that correspond to 4 and 5). This means that the client (Product Owner) is involved in the project to write the user stories. Besides, there must be support from the management (from the client's organization) to allow employees to participate and commit to the project. In case that is not true, a conflict is detected.

Rule example 3: When team members have little experience working together on agile projects (the value of the *TeamPreviousCooperation* characteristic is "Never"), the Kanban/Scrum board should not be digital.

An optional phase of the algorithm is to recommend additional candidate practices based on the specific characteristics of the target profile. To this end, a series of SPARQL queries is defined. These queries materialize the experts' knowledge about the ASD domain. When a performed SPARQL query for a practice returns a result, this means that the target profile satisfies certain characteristics required by the practice. Due to lack of space, just one example is shown, omitting the PREFIX definition. More SPARQL queries to improve recommendations can be found in [Roldan et al. 2022].

SPARQL Query 1: If the team of the target profile has more than 10 members (time size characteristic has value “10+(too large)”), the “ScrumOfScrums” practice is added to the set of candidate practices. The goal is to divide the team into smaller groups.

```
SELECT ?profile ?value WHERE {?profile rdf:type ag:TargetAgileProfile ;
                    ag:hasTeam ?team .
  ?team   ag:hasTeamCharacteristic ?teamch .
  ?teamch rdf:type ag:TeamSize ;
                    ag:hasValue ?value .
Filter (?value=<http://www.semanticweb.org/utn/AgileOntology#10+(toolarge)>)}
```

SPARQL Query 2: If the team characteristic “Turnover” of the target profile has a value between 3 and 5 (medium, medium-to-high, or high-rate), it is advisable to do short sprints to allow team members to get to know each other quickly. Therefore, the practice "PersonalMaps" is added to the set of candidate practices. This practice can also be recommended when the members of the team of the target profile have little previous cooperation or poor domain/technology knowledge. Therefore, the SPARQL query is defined as a UNION that comprises all the situations.

6. Evaluation

We have started some activities regarding the *Demonstration* and *Evaluation* steps of the research method. The first activity was mainly focused on validating the quality of the AgiPrOnt ontology. We used an automated ontology evaluation tool named OOPS! (OntOlogy Pitfall Scanner) to verify the quality of the ontology, which helps to detect some of the most common pitfalls when developing ontologies. We evaluated the AgiPrOnt ontology (at Tbox level) by submitting it to OOPS!. Some pitfalls were detected by OOPS!, which were considered to improve the ontology. On the other hand, to validate the satisfaction of the ontology requirements, we verified the coverage of the competency questions that were identified in Section 2. To prove that the requirements were satisfied, a set of SPARQL queries were performed to answer the competency questions. Due to lack of space, the queries are provided in [Roldan et al. 2022]. In addition, AgiPrOnt has been informally reviewed by human experts during its development. The reviewers were some of the organizations, which have participated in previous research studies carried out by the authors and answered the questionnaire. Several validation activities were carried out during working technical meetings and workshops. The resulting comments led to a continuous ontology improvement in order to be functionally integrated into the recommender system.

Regarding the evaluation of the recommendation algorithm, we generated 12 test cases for different scenarios, each one representing an agile profile with a particular configuration of Team/Project/Context characteristics, with clear differences between the cases. Six of them were obtained from collaborating organizations that provided actual situations prior to the start of an ASD project. The recommender provided a different set of practices for each case. However, we note that in 3 cases, the set of practices recommended was insufficient and very basic, as the ontology lack of enough data to give a better result. Despite this limitation, the performed tests were useful for adjusting the parameters of FP-growth, such us support, frequency, minimum support and frequency, minimum and maximum items per itemset, in order to obtain the best result.

The validation of the proposed recommender has other limitations. Although several tests and recommendations were conducted, it is not possible to obtain feedback in the short term on user satisfaction regarding the usefulness of the recommendations. Until the projects in which the recommended practices were adopted have not been completed, the degree of success achieved cannot be known. Finally, the knowledge represented in AgiPrOnt is still limited; with only 47 case studies, there are situations that the model cannot answer. To address this limitation, we plan to add more knowledge to the ontology in the future.

6. Conclusion

This research proposes an ontology-based recommender system of agile practices. The ontology enables the representation of agile software development knowledge so that the experiences on agile adoption can be reusable and systematic. The original approach of the Agile Practice Recommender resides in the combination of an ontology, data mining techniques, and semantic rules to detect conflicts in the recommended practices. The representation of expert knowledge using SWRL rules and SPARQL queries is useful to compensate for the deficiencies of datamining techniques applied to a small number of cases. For future work, we are working on integrating the recommender with agile development frameworks like Jira and SonarQube. Such integration would make it possible to recommend the practices at the beginning of a new project and record the ones adopted, some of which can be supported by the framework in use. In addition, the recommender could be fed with the metrics generated by these frameworks, which can help to automatically characterize the agile profiles (regarding team, project, and environment), record experiences, and add new cases to the recommender.

References

Agile Alliance ([S.d.]). Agile Glossary and Terminology. https://www.agilealliance.org/agile101/agile-glossary/, [accessed on Apr 1].

Alcaraz Calero, J. M., Marín Pérez, J. M., Bernal Bernabé, J., et al. (1 nov 2010). Detection of semantic conflicts in ontology and rule-based information systems. *Data and Knowledge Engineering*, v. 69, n. 11, p. 1117–1137.

Borges Ruy, F., Falbo, R. de A., Barcellos, M. P., Costa, S. D. and Guizzardi, G. (2016). SEON: A software engineering ontology network. In *Lecture Notes in Computer Science (LNAI volume 10024)*. Springer Verlag.

Burke, R. (2000). Knowledge-Based Recommender Systems. *ENCYCLOPEDIA OF LIBRARY AND INFORMATION SYSTEMS*, v. 69, p. 2000.

Campanelli, A. S., Camilo, R. D. and Parreiras, F. S. (1 mar 2018). The impact of tailoring criteria on agile practices adoption: A survey with novice agile practitioners in Brazil. *Journal of Systems and Software*, v. 137, p. 366–379.

Digital.ai (2021). 15th Annual State Of Agile Report. https://digital.ai/resource-center/analyst-reports/state-of-agile-report, [accessed on Mar 29].

Guizzardi, G. (2005). Ontological foundations for structural conceptual models. Telematica Instituut / CTIT.

Isinkaye, F. O., Folajimi, Y. O. and Ojokoh, B. A. (2015). Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*. Elsevier.

Júnior, P. S. S., Barcellos, M. P., Falbo, R. de A. and Almeida, J. P. A. (17 mar 2021). From a Scrum Reference Ontology to the Integration of Applications for Data-Driven Software Development. *Information and Software Technology*, p. 106570.

Kiv, S., Heng, S., Kolp, M. and Wautelet, Y. (21 may 2019). Agile methods knowledge representation for systematic practices adoption. In *Lecture Notes in Business Information Processing*. . Springer Verlag.

Kurapati, N., Manyam, V. S. C. and Petersen, K. (2012). Agile software development practice adoption survey. In *Lecture Notes in Business Information Processing*. . Springer Verlag.

Letelier, P. and Penades, M. C. (2017). AgileRoadmap: An Approach to Implement Agile Practices in Teams. *IEEE Latin America Transactions*, v. 15, n. 7, p. 1295–1300.

Meyer, B. (2014). *Agile!* Springer International Publishing.

Minoli, M. (2011). Marco de Soluciones basado en Ontologías para Desarrollo Global Ágil. Universidad Rey Juan Carlos.

Muhairat, M., ALZu'bi, S., Hawashin, B., Elbes, M. and Al-Ayyoub, M. (jan 2020). An Intelligent Recommender System Based on Association Rule Analysis for Requirement Engineering. *Journal of Universal Computer Science*, v. 26, n. 1, p. 33–49.

Nishijima, R. T. and Dos Santos, P. D. J. G. (27 jul 2013). The challenge of implementing Scrum agile methodoloy in a traditional development environment. *International Journal of Computers & Technology*, v. 5, n. 2, p. 98–108.

Oliveira Bringuente, A., Almeida Falbo, R. and Guizzardi, G. (2011). Using a foundational ontology for reengineering a software process ontology. *Journal of Information and Data Management*, v. 2, n. 3, p. 511–511.

Ortega-Ordoñez, W. A., Pardo-Calvache, C. J. and Pino-Correa, F. J. (1 apr 2019). Ontoagile: An ontology for agile software development processes. *DYNA (Colombia)*, v. 86, n. 209, p. 79–90.

Perkusich, M., De Almeida, H. O. and Perkusich, A. (2013). A model to detect problems on scrum-based software development projects. In *Proceedings of the ACM Symposium on Applied Computing*. . ACM Press.

Pinto, N., Tortosa, N., Cabas Geat, B., Ibanez, L. and Bollati, V. (26 dec 2018). Quality evaluation of agile processes: Measurement of requirements management using AQF v2. In *Proceedings - 2018 QUATIC 2018*. IEEE.

PMIBA, C. Á. (2020). 2° Estudio sobre el Pulso de la Agilidad en Argentina 2020.

Roldan, M. L., Bollati, V. and Gonnet, S. (2022). Agile practices adoption by Argentine software organizations. Mendeley Data, V3, doi: 10.17632/3chkfjxxk5.3

Stray, V., Memon, B. and Paruch, L. (2020). A Systematic Literature Review on Agile Coaching and the Role of the Agile Coach. *LNCS (subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 12562 LNCS, p. 3–19.

Tarus, J. K., Niu, Z. and Mustafa, G. (1 jun 2018). Knowledge-based recommendation: a review of ontology-based recommender systems for e-learning. *Artificial Intelligence Review*, v. 50, n. 1, p. 21–48.