Using ODA Method and FOIL Algorithm to Determine Organizational Agility Level

Gusts Linkevics, Uldis Sukovskis
Institute of Applied Computer Systems
Riga Technical University
Riga, Latvia
e-mail: gusts.linkevics@rtu.lv, uldis.sukovskis@rtu.lv

Abstract — **Agile software development has become more common during the past decade. Transitioning an organization to be agile is not an undemanding task, and it requires an active involvement of all the stakeholders. The main issue is that organizations are not aware of their agility level and the necessary operations they should perform to become more agile. The purpose of this research is to use Agility Impact Index (AII) in combination with Organization Agility Model (OAM), question generation algorithm and First-Order Inductive Learner (FOIL) algorithm to calculate the agility level of an organization. The result of the research is a proof of the Organization Domain Agility (ODA) method concept. The intention of the ODA method is to determine the agility level of a Software Development Company (SDC), which is an important step to improve the agility level of an organization.**

Keywords - Agile; Software development; FOIL; AII.

I. Introduction

Agile software development is being implemented by an increasing number of Software Development Companies (SDCs). SDCs use various agile methods, for example, Extreme Programming or Scrum [3]. There are two common issues that SDCs encounter during agile method application. One of the issues is related to the lack of awareness about their efficiency in applying agile methodology. The other common issue is the incapability to identify the exact problem in the agile methodology implementation process. Such problems can be solved by hiring agile experts or by training internal employees, which may require a significant amount of time and financial resources. The other approach is to use a method which helps to solve this problem by determining the problematic areas. Organization Domain Agility (ODA) [1] is one of the methods used for a periodical evaluation of the organisation and the team to determine the problematic areas.

The main focus of the paper is to provide a compendious introduction of the ODA method and to present detailed findings about its main components which enable to determine the agility level of an organization:

- Agility Impact Index (AII);
- Question generation algorithm;
- Domain, Sub-domain and attribute Value Tree (DSA Value Tree);
- First level rule generation using FOIL [6] algorithm.

This paper consists of 6 sections. In Section 1 the problem of the organization agility and the goal of the paper is described. In Section 2 the ODA method and its process is introduced. The ODA method is developed to evaluate the agility level of SDCs. In Section 3 the Question generation process is explained. Generated questions are used to gather the data about SDCs. In Section 4, the DSA Value Tree is depicted, and it consists of AII values. AII value is defined for each element of the DSA tree. AII values are determined by the group of experts. In Section 5, we focus on FOIL algorithm usage for the evaluation of the SDC agility level. Section 6 concludes the paper and provides an outline of the future work.

II. ODA Method

ODA method uses Organization Agility Model (OAM) [1] to describe an organization and its team in a structured way. The structural approach provides an opportunity to evaluate various parts of the organization. OAM is organized in a tree structure where the organization is described by the DSA Value Tree. The DSA Value Tree groups similar items of the organization into domains and sub-domains.

The initial model consisted of five domains [2]: Organization, Productivity, Process, Quality and Value. During the more detailed research it was noticed that an additional top level domain is needed to describe the project component of the organization (Figure 1).

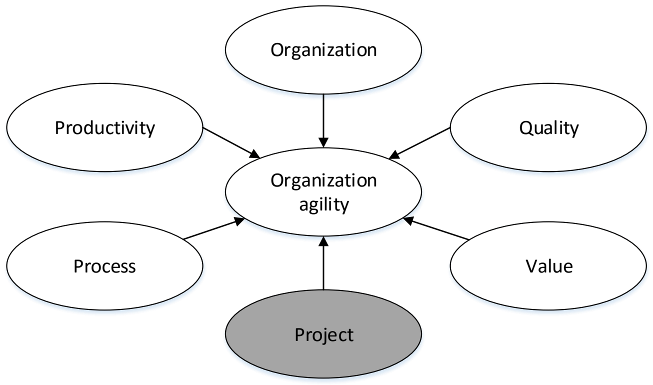

Figure 1. Extended OAM .

The purpose of the Project domain is to describe attributes of the particular project. Different projects in the SDC can be at different agility levels and can influence the Organizational agility differently.

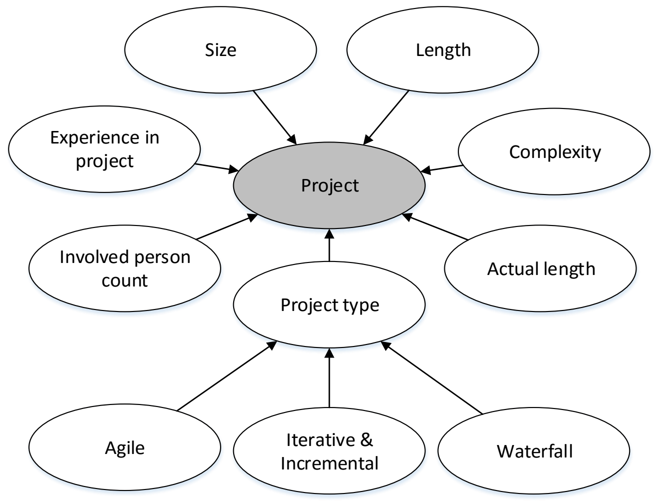

Figure 2. Project Domain Components.

Figure 2 shows the structure of the Project domain, and it consists of seven sub-domains:

- Project type – it is possible to identify 3 project subtypes:
    - Waterfall – this is a classical approach to software development.
    - Iterative and Incremental – this approach is a custom type of Agile methodology.
    - Agile – a project is based on Scrum or a similar method.
- Number of people involved – agile methods work better with small number of people within a team. In case of large projects, there is some overhead in managing the large teams, and it can decrease the overall agility level of the project.
- Experience in a project – experience of the team with the particular project also influences the agility level of the team and organization. In case the project has been running for several years and it is necessary to switch it to the agile development approach, some experienced employees may resist using the agile method in the project.
- Size – smaller projects are easier to shift to the agile approach than larger projects.
- Length – similarly to the size of the project the length of the project also will influence the agility level of the organization.
- Complexity – sometimes it is related to the project size. Complex projects fail more frequently than less complex projects as the complex projects require more formal approach. The approach could still be agile, but the more complex projects require more detailed documentation.
- Actual length (Age) – the time the project has been already running. As mentioned before, projects running for a long period of time have a potentially higher risk when switching to the agile approach than starting a new project completely with agile from the beginning.

There are several attributes that describe each sub-domain. The attribute values for the Complexity sub-domain are shown in TABLE I, and the Size (Amount of the investment) sub-domain attributes are listed in TABLE II. The list of all attributes is not included in the paper due to the limited space.

TABLE I. COMPLEXITY SUB-DOMAIN ATTRIBUTES

Attribute	Description
Low	Project is simple and does not have complex integrations and components.
Medium	Project has some integrations, but they are not complex. Project has several components which need to be integrated.
High	Complex algorithms, integrations and components are used.

TABLE II. SIZE SUB-DOMAIN ATTRIBUTES

Attribute	Investment amount
Enhancement	x < $250,000
Small	$250,000 < x < $ 1M
Medium	$ 1M < x < $ 3M
Large	$ 3M < x < $ 10M
Very large	x > $ 10M

AII values range from 1 to 10, where 1 means that the DSA item does not influence organization agility, and 10 means that the item significantly impacts the agility, for example, the Productivity domain does not influence the agility level the same way as Organization domain. AII is determined by the expert evaluation method DELPHI [5] which uses an external agile expert network to evaluate the common DSA. Agile experts do not need any information about the particular organization, and they are not directly related to it. The evaluation they provide is bound to the common DSA, which is then used together with the information acquired from the particular organization. In general, there is a basic agile knowledge which can be applied to any organization looking towards agile software development.

Agile experts evaluate the DSA at least once and then repeat it if the structure of the DSA is changed. It is possible to change the DSA structure for the ODA method (Figure 3.) in case the organization uses any other agile method than Scrum.

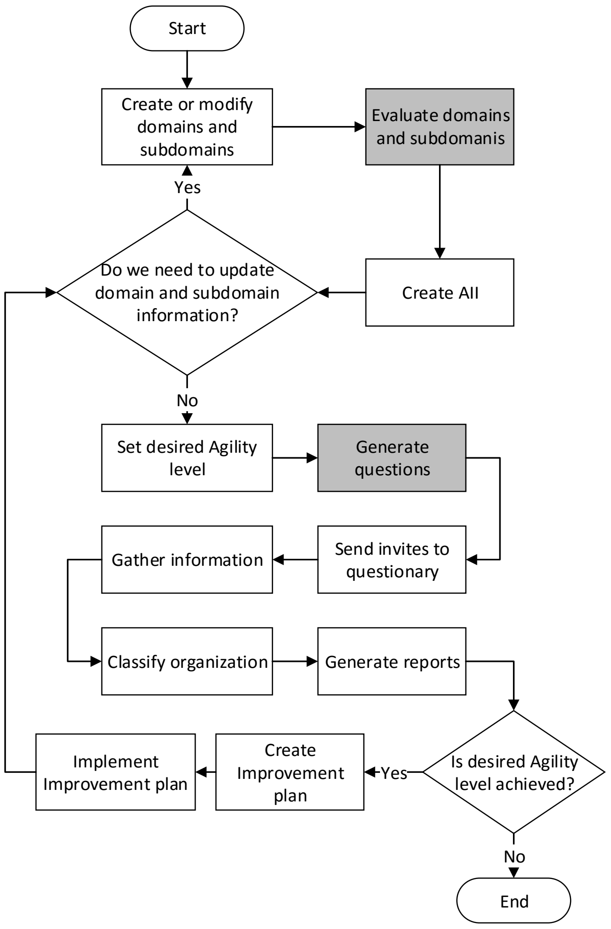
Start
Create or modify domains and subdomains
Evaluate domains and subdomanis
Yes
Do we need to update domain and subdomain information?
Create AII
No
Set desired Agility level
Generate questions
Gather information
Send invites to questionary
Classify organization
Generate reports
Implement Improvement plan
Create Improvement plan
Yes
Is desired Agility level achieved?
No
End

Figure 3. Process of a Method ODA.

After all of the AII values for the DSA are determined, the Question generation algorithm is used to generate question sets for the employees. In the next section, the Question generation algorithm is described in more detail. The generated questions are distributed among employees in different departments.

III. Question Generation

Question generation is an important part of the ODA method, and it is required to generate only a small set of questions for each employee at each evaluation period. There are approximately 300 questions to ask, and it is impractical to ask each employee all of them. Based on the question ranking by AII value, the most influential questions are asked first.

As it is shown in Figure 4 the Question generator generates subsets of questions from the set of all questions (1).

$$Q = \{q_1, q_2, q_3 \dots q_m\} \tag{1}$$

Where:

- Q – Set of all questions.
- $q_{1\dots m}$ – Questions, where m is the total number of questions.

An employee based question set can be defined as a subset of all questions (2).

$$A_{1\dots n} \in Q \tag{2}$$

Where:

- Q – Set of all questions.
- $A_{1\dots n}$ – Subsets of questions for an employee, where n is the amount of employees participating in survey.

The organization sets the number of questions for each employee. It is not recommended to create large sets of questions as it may lead to low quality of answers. It is recommended to include up to 10 questions in each evaluation [8], and the evaluation process should take from 5 to 7 minutes.

There are three types of questions in the employee question set (3):

- Priority questions – the initiator of question generation marks a number of questions to be included in all of the generated question sets. The priority questions make 20 per cent of all the questions in the set.
- Unanswered questions ordered by AII – a list of all the unanswered questions ordered by AII value. After adding High priority questions to the set, the unanswered questions are added to the set. This is required to cover the maximum amount of information about the DSA. At the beginning the most influential questions are added. Those questions make 60 per cent of all the questions in the set.
- Previously answered questions ordered by AII – this type of questions helps to keep the “pulse” on the most influential DSA elements, and those questions form 20 per cent of all the questions.

$$A_{1\dots n} = \{P_q, N_{1\dots n}, O_{1\dots n}\}. \tag{3}$$

Where:

- $A_{1\dots n}$ – Question set for particular employee.
- P_q – Set of priority questions.
- $N_{1\dots n}$ – Unanswered questions ordered by AII.
- $O_{1\dots n}$ – Previously answered questions ordered by AII.

The question generation process is shown in Figure 4. For example, if there are 17 questions in the question set Q, as in

$$Q = \{q_1=9, q_2=8, q_3=7, q_4=6, q_5=9, q_6=8, q_7=7, q_8=7, q_9=8, q_{10}=9, q_{11}=9, q_{12}=8, q_{13}=7, q_{14}=7, q_{15}=6, q_{16}=7, q_{17}=8\}. \tag{4}$$

And 4 questions in priority question set P_q (Priority questions have been selected by generation initiator) as in

$$P_q = \{q_{10}=9, q_{17}=8, q_3=7, q_{15}=6\}. \tag{5}$$

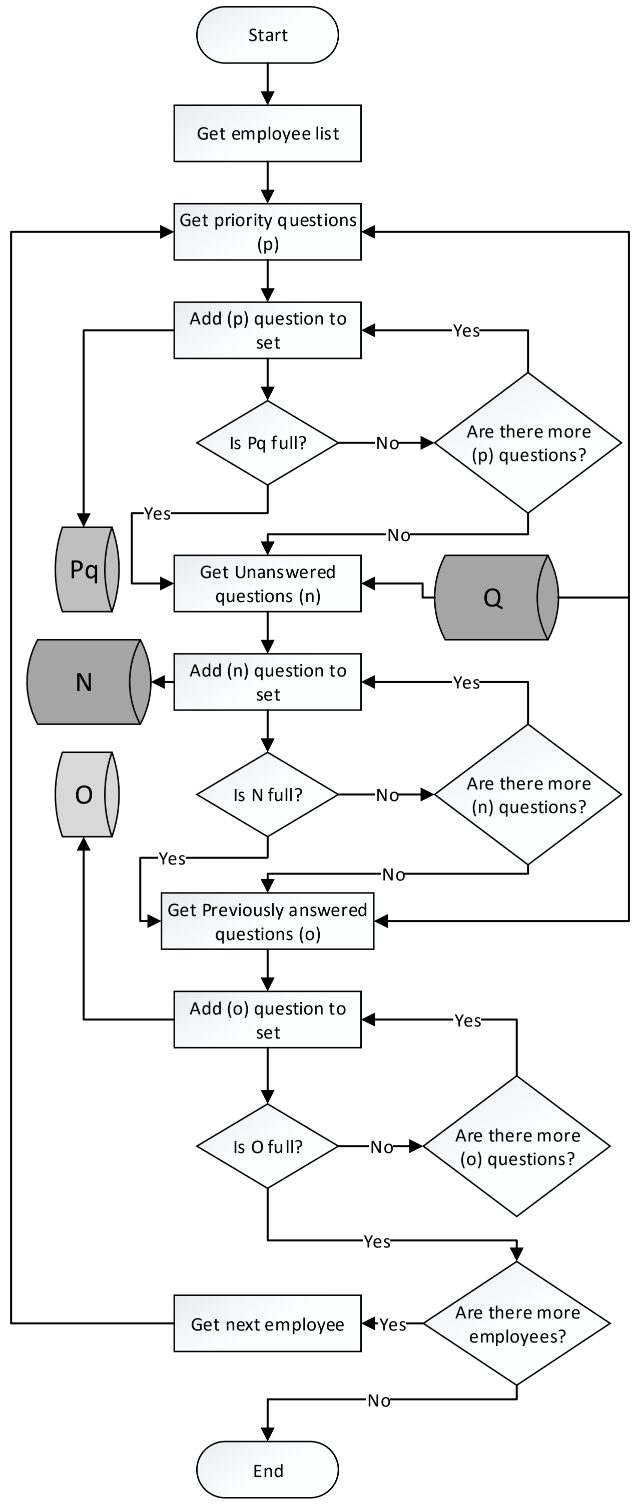
Start
Get employee list
Get priority questions (p)
Add (p) question to set
Is Pq full?
No
Are there more (p) questions?
Yes
Yes
No
Pq
Get Unanswered questions (n)
Q
N
Add (n) question to set
Is N full?
Are there more (n) questions?
O
Get Previously answered questions (o)
Add (o) question to set
Is O full?
Are there more (o) questions?
Get next employee
Are there more employees?
End

Figure 4. Question Generation Process.

And 4 questions in the previously answered question set O, as in

$$O = \{q_{12}=8, q_7=7, q16=7, q_4=6\}. \quad (6)$$

And there is one employee involved in the evaluation of 10 questions, then the resulting question set would contain questions

$$A_1 = \{P\{q10,q17\},N\{q1,q5,q11,q2,q6,q9\},O\{q12,q7\}\}. \quad (7)$$

Depending on the number of employees correct timing for question generation should be selected. As question generation depends heavily on question sets N and O, then it is reasonable to assume that the questions are generated during the night. In this way, also the system is not congested during working hours.

After question generation, question sets are sent to each employee. Time for question sending should be selected properly [9], in this case, after the Review meeting and before the Retrospective meeting. The gathered information is used to build the DSA Value Tree.

IV. DSA VALUE TREE

DSA value tree is a way to represent the gathered data. Tree view is a convenient way to identify the problematic areas and compare the gathered data with the AII values identified by the expert.

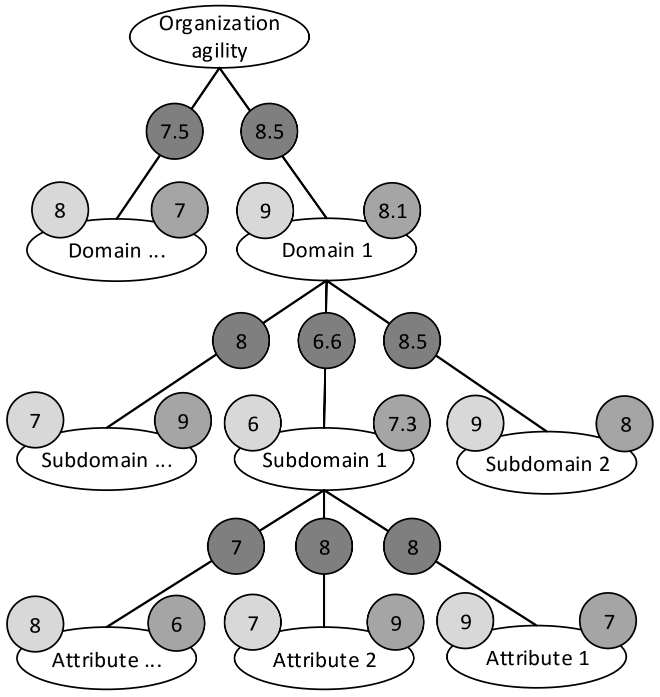

Figure 5. DSA value tree example.

Figure 5 shows a sample of a DSA Value Tree where the values on the right are the values gathered from the employee surveys (ESV), and the values on the left are the AII values defined by the experts. ESV values are calculated using average weighted values (5).

$$\bar{y} = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i} \tag{8}$$

Where:

- ȳ – Set of all questions.
- w_i – weight value, in this case AII.
- x_i – ESV value, determined from surveys.
- n – Number of respondents who have answered particular question.

The agility level can be determined on an organization level, on a project level or on a team level. The following breakdown is required to improve the agility level at a particular team or a project. As mentioned before, the agility level can differ on a team or a project level. In case of a team or a project agility level determination, filtering of set ȳ is used to include only ESV values for a particular team or a project (Figure. 6).

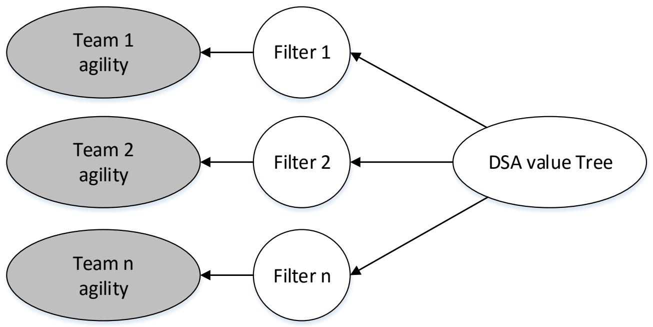

Figure 6. DSA Value Tree filtered for the team.

One of the ways to represent the gathered data is grouping the data by project and team after the DSA value tree is created (Figure. 7).

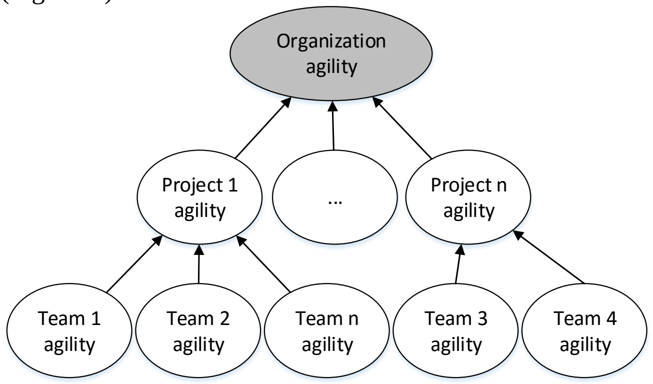

Figure 7. Grouping of the DSA value tree.

The grouping approach helps to identify the problematic projects and teams where additional effort for the agility level increment is needed.

After all ȳ values are calculated, the rules based on these values can be created. The ODA method uses FOIL algorithm to generate the rules for the final evaluation of the SDC agility level.

V. Using FOIL Algorithm For The Rule Generation

FOIL is a system for finding function-free Horn clauses [6]. FOIL searches for the first-order rules using a learning set. The search results in finding a set of logical rules describing the system under consideration.

The first-order rule is a logical proposition of the form:

$$R(V1,V2,\ldots,Vk) \leftarrow L_1, L_2, \ldots, L_m, \tag{9}$$

Where:

- R is a target relation between variables V_i.
- L_i are literals composing a condition which verity enables one to state that the head of the rule is true.

Within this research FOIL is used to determine the agility level of the SDC where the set of the first-order rules determines the agility class of the SDC. Before generating the first-order rules for determining the agility level of an Organization, a Project or a Team, it is necessary to define the agility classes. The Agility class determines the present agility level of an organization, a project or a team. Knowledge helps to create a specific improvement plan and to implement it later. It is reasonable to use 5 or 10 agility classes of evaluation, as it is with grades at school. Some school systems use 5 point grading system, whereas some schools use 10 point grading system. It depends on how accurately we want to evaluate. In this case, it is decided to use 5 agility classes (K1, K2, K3, K4 and K5) which will map to the average values of the DSA Value Tree. Each class corresponds to 2 values (Table III).

TABLE III. Agility Class Mapping to the DSA Value Tree

DSA Value Tree values	Agility Class	Description
1, 2	K1	Not agile and no evidence of agility
3, 4	K2	Not agile, but some evidence of agility exists
5, 6	K3	Some evidence of agility, but major improvements should be introduced
7, 8	K4	Agile, but some problems exist and requires some improvements
9, 10	K5	Agile and no important improvements are needed.

The first-order rules are important for the agility class determination because it is not possible to determine the exact level of agility there is only the information about the DSA Value Tree. For example, if the top-level domain average values are 1, 4, 5, 6, 7 and 8 (each value represents average value of the Organization, Productivity, Quality, Project, Value and Process domain), one should analyse in more detail if it means the organization is agile. The same question may be discussed if the DSA Value Tree values are 5, 2, 7, 9, 3 and 8. The FOIL algorithm can be used to resolve such problems.

FOIL algorithm uses learning data set which contains information about the known organisations valued by experts. The learning data set has information about the specific organisation values and its class. There is a small learning data sample presented in Table IV. The data listed in the Table IV has been simplified to shorten the solution (Value ranges are shortened and not all the domains are included). Columns D1, D2, D3 and D4 represent four top level domains Organization, Process, Productivity and Quality. The sample learning data set contains learning data only for three agility classes N1 (Not agile and no signs of agility), N2 (Some signs of agility, but major improvements should be introduced) and N3 (Agile and no important improvements are needed) and is also simplified.

TABLE IV. SAMPLE FOIL LEARNING DATA

D1	D2	D3	D4	Class
1	1	1	1	N1
1	1	1	2	N2
1	1	2	1	N1
1	1	2	2	N3
1	2	1	1	N1
1	2	1	2	N2
1	2	2	1	N1
1	2	2	2	N3
3	1	1	1	N1
3	1	1	2	N2
3	1	2	1	N1
3	1	2	2	N3
3	2	1	1	N1
3	2	1	2	N2
3	2	2	1	N1
3	2	2	2	N1
2	1	1	1	N1
2	1	1	2	N1
2	1	2	1	N1
2	1	2	2	N3
2	2	1	1	N1
2	2	1	2	N2
2	2	2	1	N1
2	2	2	2	N1

First-order rules can be described in a form of IF … THEN … or in a form of Horn clauses, Head ← Body [6]. In this case Agility Class ← Condition. In case of simplified learning data there are three agility classes N1, N2 and N3. Each class in the learning data set has a specific number of records N1 = 15, N2 = 5 and N3 = 4. The learning data set is used to teach the algorithm how to identify particular agility class.

FOIL algorithm uses FOIL_GAIN function to evaluate each next literal to be added to the class identification rule (10) [6].

$$Foil\_Gain(L, R) \equiv t \left(\log_2 \frac{p_1}{p_1 + n_1} - \log_2 \frac{p_0}{p_0 + n_0} \right) \quad (10)$$

Where:

- L – New condition to be added to the rule.
- R – Rule body, to which we want to add the condition.
- p_0 – Number of positive items in rule R.
- n_0 – Number of negative items in rule R.
- p_1 – Number of positive items in rule R_1.
- n_1 – Number of negative items in rule R_1.
- T – Number of positive items in rule R after adding new condition L to the rule R.

To create rules for the class N1 we need to identify all positive and negative examples, as seen in Table V and Table VI.

TABLE V. POSITIVE SAMPLES FOR CLASS N1

D1	D2	D3	D4	Class
1	1	1	1	N1
1	1	2	1	N1
1	2	1	1	N1
1	2	2	1	N1
3	1	1	1	N1
3	1	2	1	N1
3	2	1	1	N1
3	2	2	1	N1
3	2	2	2	N1
2	1	1	1	N1
2	1	1	2	N1
2	1	2	1	N1
2	2	1	1	N1
2	2	2	1	N1
2	2	2	2	N1

TABLE VI. NEGATIVE SAMPLES FOR CLASS N1

D1	D2	D3	D4	Class
1	1	1	2	N2
1	1	2	2	N3
1	2	1	2	N2
1	2	2	2	N3
3	1	1	2	N2
3	1	2	2	N3
3	2	1	2	N2
2	1	2	2	N3
2	2	1	2	N2

During the next step a new literal should be added (11).

$$I(T_1) = -\log_2(15/(15+9)) = 0{,}678 \quad (11)$$

After checking the results (Table VII) a literal D4(X, 1) can be added to the rule N1 ← D4(X, 1).

TABLE VII. CALCULATION OF FOIL_GAIN FOR CLASS N1

Literal	Calculation	Foil_Gain
D4(X, 1)	$I(T_2) = -\log_2(12/(12+0)) = -\log_2(1)=0$	12*(0,678-0) = 8,136
D4(X, 2)	$I(T_2) = -\log_2(3/(3+9)) = -\log_2(0,25)=2$	3*(0,678-2) = -3,966
D3(X, 2)	$I(T_2) = -\log_2(8/(8+4)) = -\log_2(0,666)= 0,586$	8*(0,678-0,586) = 0,736
D3(X, 1)	$I(T_2) = -\log_2(7/(7+5)) = -\log_2(0,583)= 0,788$	7*(0,678-0,788)=- 0,77
D2(X, 1)	$I(T_2) = -\log_2(7/(7+5)) = -\log_2(0,583)= 0,788$	7*(0,678-0,788)=- 0,77
D2(X, 2)	$I(T_2) = -\log_2(8/(8+4)) = -\log_2(0,666)= 0,586$	8*(0,678-0,586) = 0,736
D1(X,1)	$I(T_2) = -\log_2(4/(4+4)) = -\log_2(0,5)= 1$	4*(0,678-1) = -1,288
D1(X,3)	$I(T_2) = -\log_2(5/(5+3)) =$	5*(0,678-0,678) = 0

Literal	Calculation	Foil_Gain
	$-\log_2(0{,}625)= 0{,}678$	
D1(X,1)	$I(T_2) = -\log_2(6/(6+2)) = -\log_2(0{,}75)= 0{,}415$	2*(0,678-0,415) = 0,263

As this condition does not return any negative samples, it is possible to stop the further processing of the rule. However, as this rule does not return all the positive samples, it is necessary to add an additional condition to the rule set for class N1. After removing all the positive samples covered by the first condition, there are only 3 positive samples left (Table VIII).

TABLE VIII. POSITIVE SAMPLES FOR CLASS N1, AFTER REMOVING POSITIVE SAMPLES AFFECTED BY FIRST CONDITION

D1	D2	D3	D4	Class
3	2	2	2	N1
2	1	1	2	N1
2	2	2	2	N1

During the following step the next literal should be added (12).

$$I(T_1) = -\log_2(3/(3+9)) = 2 \qquad (12)$$

TABLE IX. CALCULATION OF FOIL_GAIN FOR CLASS N1

Literal	Calculation	Foil_Gain
D4(X, 2)	$I(T_2) = -\log_2(3/(3+9)) = -\log_2(0{,}25)=2$	3*(2-2) = 0
D3(X, 2)	$I(T_2) = -\log_2(2/(2+4)) = -\log_2(0{,}333)= 1{,}586$	2*(2-1,586) = 0,414
D3(X, 1)	$I(T_2) = -\log_2(1/(1+5)) = -\log_2(0{,}166)= 2{,}59$	1*(2-2,59) = -0,59
D2(X, 1)	$I(T_2) = -\log_2(1/(1+5)) = -\log_2(0{,}166)= 2{,}59$	1*(2-2,59) = -0,59
D2(X, 2)	$I(T_2) = -\log_2(2/(2+4)) = -\log_2(0{,}333)= 1{,}586$	2*(2-1,586) = 0,414
D1(X,1)	$I(T_2) = -\log_2(0/(0+4)) = -\log_2(0)= \infty$	-
D1(X,3)	$I(T_2) = -\log_2(1/(1+3)) = -\log_2(0{,}25)= 2$	1*(2-2) = 0
D1(X,2)	$I(T_2) = -\log_2(2/(2+2)) = -\log_2(0{,}5)= 1$	2*(2-1) = 2

As shown in Table IX, the most notable gain is from literal D1(X, 2), which can be added to the rule, but, as it also selects negative samples, more literals should be added.

TABLE X. CALCULATION OF FOIL_GAIN FOR CLASS N1

Literal	Calculation	Foil_Gain
D4(X, 2)	$I(T_2) = -\log_2(2/(2+2)) = -\log_2(0{,}5)=1$	2*(2-1) = 2
D3(X, 2)	$I(T_2) = -\log_2(1/(1+1)) = -\log_2(0{,}5)=1$	1*(2-1) = 1
D3(X, 1)	$I(T_2) = -\log_2(1/(1+1)) = -\log_2(0{,}5)=1$	1*(2-1) = 1
D2(X, 1)	$I(T_2) = -\log_2(1/(1+1)) = -\log_2(0{,}5)=1$	1*(2-1) = 1
D2(X, 2)	$I(T_2) = -\log_2(1/(1+1)) = -\log_2(0{,}5)=1$	1*(2-1) = 1

To this point the second rule is N1 ← D1(X, 2) ∧ D4(X, 2) and it has to be checked if it returns any negative samples. Considering the fact that it returns negative samples, the additional literal should be added.

TABLE XI. CALCULATION OF FOIL_GAIN FOR CLASS N1

Literal	Calculation	Foil_Gain
D3(X, 2)	$I(T_2) = -\log_2(1/(1+1)) = -\log_2(0{,}5)=1$	1*(2-1) = 1
D3(X, 1)	$I(T_2) = -\log_2(1/(1+1)) = -\log_2(0{,}5)=1$	1*(2-1) = 1
D2(X, 1)	$I(T_2) = -\log_2(1/(1+1)) = -\log_2(0{,}5)=1$	1*(2-1) = 1
D2(X, 2)	$I(T_2) = -\log_2(1/(1+1)) = -\log_2(0{,}5)=1$	1*(2-1) = 1
D3(X, 2)	$I(T_2) = -\log_2(1/(1+1)) = -\log_2(0{,}5)=1$	1*(2-1) = 1

As shown in Table XI, in this case all the literals are equally bad as all return negative samples. As a consequence, one more literal should be added to the rule. To this point the rule is N1 ← D1(X, 2) ∧ D4(X, 2) ∧ D3(X, 2).

TABLE XII. CALCULATION OF FOIL_GAIN FOR CLASS N1

Literal	Calculation	Foil_Gain
D2(X, 1)	$I(T_2) = -\log_2(0/(0+1)) = -\log_2(0)=\infty$	-
D2(X, 2)	$I(T_2) = -\log_2(1/(1+0)) = -\log_2(1)=1$	1*(2-0) = 2

At this point literal D2(X, 2) can be added to the rule set N1 ← D1(X, 2) ∧ D4(X, 2) ∧ D3(X, 2) ∧ D2(X, 2).

Acknowledging the fact that all the positive examples are not yet covered, an additional rule should be added to the set. All the positive samples covered by the new rule should be removed from the learning set, and a search of a new rule should be continued. The process is repeated as many times as necessary until all the positive samples are covered. At the end, the rule set for class N1 looks like:

- N1 ← D4(X, 1)
- N1 ← D1(X, 2) ∧ D4(X, 2) ∧ D3(X, 2) ∧ D2(X, 2)
- N1 ← D1(X, 2) ∧ D4(X, 2) ∧ D3(X, 1) ∧ D2(X, 1)
- N1 ← D1(X, 3) ∧ D3(X, 2) ∧ D2(X, 2)

This process is repeated for class N2 and N3. A complete rule set for all tree classes is shown in Table XIII.

TABLE XIII. FINAL RULE SET FOR CLASSES N1, N2 AND N3

Class	Rule set

N1	N1 ← D4(X, 1)
	N1 ← D1(X, 2) ∧ D4(X, 2) ∧ D3(X, 2) ∧ D2(X, 2)
	N1 ← D1(X, 2) ∧ D4(X, 2) ∧ D3(X, 1) ∧ D2(X, 1)
	N1 ← D1(X, 3) ∧ D3(X, 2) ∧ D2(X, 2)

N2	N2 ← D4 (X, 2) ∧ D3 (X, 1) ∧ D2 (X, 2)
	N_2 ← D4 (X, 2) ∧ D3 (X,1) ∧ D2 (X, 1) ∧ D1 (X, 1)
	N2 ← D3 (X, 1) ∧ D4 (X, 2) ∧ D1 (X, 3)

N3	N_3 ← D4 (X, 2) ∧ D3 (X, 2) ∧ D1 (X, 1) N_3 ← D4 (X, 2) ∧ D3 (X, 2) ∧ D2 (X, 1)

When the learning data set is processed, the rule sets can be tested against the learning data set (Table XIV). In this case, rules generated by FOIL can be used to identify tree classes of agility N1, N2 and N3. These rules are simplified, but the approach can be used to generate more complex rules to satisfy the needs of the ODA method.

TABLE XIV. RULE SET TESTING RESULTS

No.	D1	D2	D3	D4	Class	Result	Rule
1	1	1	1	1	N1	N1	1
2	1	1	1	2	N2	N2	6
3	1	1	2	1	N1	N1	1
4	1	1	2	2	N3	N3	8,9
5	1	2	1	1	N1	N1	1
6	1	2	1	2	N2	N2	5
7	1	2	2	1	N1	N1	1
8	1	2	2	2	N3	N3	8
9	3	1	1	1	N1	N1	1
10	3	1	1	2	N2	N2	7
11	3	1	2	1	N1	N1	1
12	3	1	2	2	N3	N3	9
13	3	2	1	1	N1	N1	1
14	3	2	1	2	N2	N2	5,7
15	3	2	2	1	N1	N1	1,4
16	3	2	2	2	N1	N1	4
17	2	1	1	1	N1	N1	1
18	2	1	1	2	N1	N1	3
19	2	1	2	1	N1	N1	1
20	2	1	2	2	N3	N3	9
21	2	2	1	1	N1	N1	1
22	2	2	1	2	N2	N2	5
23	2	2	2	1	N1	N1	1
24	2	2	2	2	N1	N1	2

One of the problems of this approach is that there is a need for a set of quality training data for the algorithm, which is not so easy to gather.

VI. CONCLUSION

Shifting a project to agile software development is not an effortless procedure, and there are different ways to accomplish it. Some organizations hire expensive agile experts, while others try to execute the transition process themselves. The ODA method can support the transition process. It has several steps, and it starts with the creation of OAM. The next step is the DSA evaluation carried by agile experts using the DELPHI method. The evaluated AII values are later used to generate the employee-based question sets. The data gathered from the questionnaires is used to create the DSA value tree. When the DSA value tree is created, it is used by the FOIL method to generate rules for determining the agility level.

The process of assessing the actual organization agility is long, but in most cases it can be completely automated. For example, the AII values are already defined for the DSA, and the organization does not need to hire any agile experts. It could be required only in cases when the existing DSA does not match the organization, especially in the process domain part. The initial process domain of the DSA is built based on Scrum, which resulted to be the most common agile method during the last few years.

As there are approximately 300 questions in the question set Q, there is a risk to fail in collecting the necessary data from all the employees. To mitigate this risk, the Question generation algorithm is used to generate smaller subsets of questions each time. The Question set size depends on the SDC. Some organizations could generate sets of 10 questions whereas other organizations could generate 15 questions per set. The question amount per set is configurable in the supporting tool of the ODA method. The Question sets are generated periodically, and they include three types of questions. There are High priority questions which are included in the question set if it is necessary to gather the feedback within a short period of time. There are Unanswered questions ordered by AII and Answered questions with high AII values which need to be answered more frequently.

There is additional risk related to AII values. In order to make ODA method work correctly, the agile expert network should be of a high quality and expertise. High quality expert network creation is not an easy task, but it is achievable, and mostly the SDC who use the ODA method will not need to create the network themselves.

Rule generation using FOIL is automated, and the algorithm is suitable for grouping tasks. It is assumed that it is possible to use similar algorithms as well. The biggest challenge at this step is to have a good quality learning data for the algorithm as the quality of the generated rules depends on the quality of the learning data.

During the further research it is planned to test the method and the used algorithms on several organizations, as the concept of this approach proves to be beneficial.

REFERENCES

[1] G. Linkevics, “Evaluation of Agility in Software Development Company” Joint International Conference on Engineering Education & International Conference on Information Technology (ICEE/ICIT 2014), 2014, pp. 102.

[2] G. Linkevics, “Adopting to Agile Software Development”, Applied Computer Systems, 2014, pp. 64 -70.

[3] K. S. Rubin, “Essential Scrum: A Practical Guide to Most Popular Agile Process”, Addision-Wesley Professional, 2012.

[4] M. Poppendieck and T. Poppendieck, “Implementing Lean Software Development: From Concept to Cash”, Addision-Wesley Professional, 2006.

[5] G. J. Skulmoski, F. T. Hartman and J. Krahn, “The Delphi Method for Graduate Research”, Journal of Information Technology Education, vol. 6, 2007

[6] J. R. Quinlan. “Learning logical definitions from relations”, Boston: Kluwer Academic Publishers, 1963, pp. 239–266.

[7] Manifesto for Agile Software Development, http://agilemanifesto.org/, [retrieved: 12, 2014].

[8] SurveyMonkey, How Much Time are Respondents Willing to Spend on Your Survey? https://www.surveymonkey.com/blog/2011/02/14/survey_completion_times/, [retrieved: 8, 2015].

[9] Fluid Surveys University, It’s All About Timing –When to Send your Survey Email Invites? http://fluidsurveys.com/university/its-all-about-timing-when-to-send-your-survey-email-invites/, [retrieved: 8, 2015].