Identification and Definition of Early Aspects - A Prototype of Method

ANTONIO MARIA PEREIRA DE RESENDE[1]
ADILSON MARQUES DA CUNHA[2]
HEITOR AUGUSTUS XAVIER COSTA[3]

[1, 3]PqES - Software Engineering Research Group
[1, 3]Federal University of Lavras - Department of Computer Science
ZIP 37200-000, Lavras, MG, Brazil
[1]tonio@dcc.ufla.br, [3]heitor@dcc.ufla.br

[2] Technological Institute of Aeronautics
Department of Electronic Engineering and Computation
ZIP 12228-900, São José dos Campos, SP, Brazil
[2]cunha@ita.br

Abstract. This paper briefly presents the conceptual model of the Method for Early Aspect (EA) Identification and Definition (MEAID). It also details activities named Early Aspects Candidates Identification with a set of heuristics and Early Aspects Definition with decision equation. The MEAID has being developed to support software engineering professionals reduce empirical and subjective decisions, aiming to increase efficacy and efficiency on such activity. Results from a scientific experimentation, based on Experimental Software Engineering concepts and their statistical analysis applying Student's T Test statistical method are presented as well.

Keywords: Workflow process management, Dynamic access control integration, Authorization policy

(Received January 20, 2010 / Accepted May 30, 2010)

1. Introduction

The emergence of Object-Oriented (OO) paradigm shows up several benefits to the software engineering field, such as the software development complexity reduction, as well as facilities to maintain, modularize and reuse software. Despite of such contributions during the previous decades, OO seems to achieve their limits for reducing systems complexity nowadays [1], [2], [3], and [4]. The Aspect-Oriented (AO) has appeared on this context, being able to reduce the software development complexity and keeping benefits achieved by OO [3].

Along with AO, needs of figuring out a methodology of Aspect Oriented Software Development (AOSD) has emerged in order to identify, separate, design and compose aspects and crosscutting concerns This methodology shall support software engineering phases such as analysis, design, implementation, testing and maintenance.

Early Aspects (EA) represent the subset of activities belonging to the AOSD, [5] and [6] aiming to identify aspects from initial phases of software development as domain analysis, requirements specification and architectural design, as shown in Figure 1.

In a previous paper [7] and others as [8], [9], and [10], it was shown a review about Early Aspects where solution proposal were divided into Methodological and Templates ones. The former one describes a method to early aspects identification, whereas the latter one describes structures, schemas, taxonomies, patterns and others to be used as references to guide software modeling to identity early aspects. Keeping this criterion in mind, MEAID is classified as a Methodological solution.

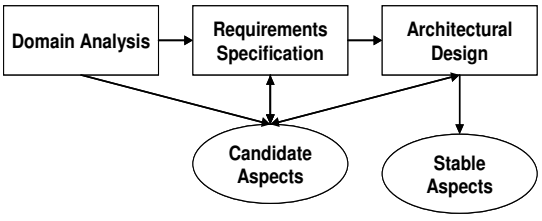

Figure 1 - Early Aspect Scope.

This paper presents a conceptual model of the Method for Early Aspect Identification and Definition (MEAID) and details the Identifying Early Aspects Candidates. It also shows up a set of heuristics to

identify them, besides to describe a experiment carried out from Experimental Software Engineering concepts.

The rest of this paper is organized as follows. Section 2 introduces MEAID. Section 3 briefly summarizes the experimentation realized and its results. Next, concluding remarks are given in Section 4.

2. MEAID

Nowadays, some researchers are investigating methods, techniques, and tools for Early Aspect. Several of them are resumed in [11], but none of them show up heuristics or Experimental analysis, applying statistics, to determine their exactly values.

The Method for Early Aspect Identification and Definition (MEAID) has two main activities: Early Aspect Candidates Identification and Early Aspects Definition.

The former one uses two artifacts as input: Requirements Specification and Heuristics to Identify Candidates Early Aspects (HIEAC). This activity enables appropriate requirements identification, showing to the software engineering that he/she might take advantage if the implementation was made by using Aspect-Oriented. The output artifact of this activity is named Specification of Early Aspects Candidates, as depicted in Figure 2, which comprises Early Aspects Candidates found out by this activity. The execution of this activity consists of applying heuristics detailed in Table 1, according to the Requirements Specification of the software under development.

Table 1 - Heuristics Description.

HEURISTICS FORM	
HIEAC-1	***Rule*:** If there are Non-Functional Requirements (NFR) of the same nature where they can be grouped into modules, packages or components, then each set of this requirements is an EA Candidate (EAC).
	***Examples*:** • If NFR related to Safety can be grouped into modules, packages or components, then each one of these sets of the same nature is an EAC; • If NFR related to Distribution can be grouped into modules, packages or components, then each one of these sets of the same nature is an EAC; • If NFR related to Persistence can be grouped into modules, packages or components, then each one of these sets of the same nature is an EAC; • If NFR related to Transactions can be grouped into modules, packages or components, then each one of these sets of the same nature is an EAC; • If NFR related to Concurrency can be grouped into modules, packages or components, then each one of these sets of the same nature is an EAC; and • If NFR related to Synchronization can be grouped into modules, packages or components, then each one of these sets of the same nature is an EAC.
HIEAC-2	***Rule*:** If there are NFR identified by HIEAC-1 which can be subdivided into smaller parts, then each one of these subdivisions is an EAC.
	***Examples*:** • If NFR related to Safety can be subdivided into smaller parts, as Authentication, Authorization, Logging, Cryptography and others, then each one of these subdivisions is an EAC.
HIEAC-3	***Rule*:** If there are programmable requirements by means of different technologies, then each one of them is an EAC
	***Examples*:** • If Data Transmission requirements can be programmed using Hypertext Transfer Protocol - HTTP, User Datagram Protocol - UDP, Sockets, Transmission Control Protocol / Internet Protocol - TCP/IP, among others, then these requirements are EAC; • If Distribution requirements can be programmed using Common Object *Request Broker Architecture - CORBA*, *Remote Method Invocation - RMI* or *Distributed Component Object Model - DCOM*, then these requirements are EAC; and • If Persistence requirements can be programmed using *Prevailed*, *Hibernate*, *Java Data Objects JDO* or *Enterprise Java Beans - EJB*, then these requirements are EAC.

Table 1 - Heuristics Description (cont.).

HEURISTICS FORM	
HIEAC-4	***Rule*:** If there are requirements related to system monitoring to fault detection or fault tolerance, then each one of these requirements is an EAC.
	***Examples*:** • Requirements related to Pre-Conditions - if there are requirements which apply pre-conditions, then each one of them is an EAC. • Pre-Condition Example: a) All banking transactions must be encrypted; b) All credit and debit operations using internet, besides to ask for id and password for internet access, must ask for id and password used in banking house; e c) It must have at least one registered employee to generate payroll. • Requirements related to Post-Conditions - if there are requirements which apply post-conditions, then each one of them is an EAC. • Post-Condition Example: a) All employees must have an address and a phone number registered; and b) Employees must be registered with at least 2 minimum salary and medical care coverage. • Requirements related to Warranty Restrictions - if there are requirements which apply warranty restrictions, then each one of them is an EAC. • Warranty Restriction Example: a) If the pressure is over 2000 lbs, set it to 2000; and b) If there is any employee without a manager, associate to them managers which have fewer employees under supervision. • Requirements related to Exceptions Treatment - if there are requirements which apply exceptions treatment, then each one of them is an EAC. • Exceptions Treatment Example: a) If there is any calculus error, then open an dialog and ask for values to assign to the variable which has presented the error; and b) If the sensor´s visibility is under 15 meters, slow down the speed to under 30 km/h.
HIEAC-5	***Rule*:** If there are requirements related to other requirements pointing out coupling functionalities, then each one of these requirements observed at least once is an EAC. PS.: A special care must be taken regarding verbs and nouns which might indicate the same functionality or requirement, even when they are different. The opposite idea is true as well.
	***Examples*:** • If Safety requirements appear in another requirements and are written in a different way, for instance: a) Authenticate a user; and b) Check user id and password, then Safety is an EAC. • If Communication requirements appear in another requirements and are written in an identical way, such as: a) Send order to headquarters office; b) Send final balance of the day to headquarters office; and c) Send list of products broken to registration on stock; then Communication is an EAC.
HIEAC-6	***Rule*:** If there are a set of requirements that can be gathered and implemented as a component to be reused in others projects, then that requirements set is an EAC.
	***Examples*:** • If the requirements set responsible for automatic riot detection using video can be gathered and implemented as a component to be reused in others projects, then that requirements set is an EAC. • If the requirements set responsible for oil pipeline leak detection can be gathered and implemented as a component to be reused in others projects, then that requirements set is an EAC. • If the requirements set responsible for authentication and authorization users can be gathered and implemented as a component to be reused in others projects, then that requirements set is an EAC.
HIEAC-7	***Rule*:** If there are requirements available in different software editions (eg. home, professional and enterprise) depending on customer interest, then each one of these requirements is an EAC.
	***Examples*:** • If the requirements set responsible for automatic riot detection using video cannot be implemented in home edition, but it can be implemented in professional edition, depending on customer´s choice, and it is implemented by default in enterprise edition, then each one of these requirements is an EAC. • If the requirements set responsible for oil pipeline detection cannot be implemented in home edition, but it can be purchased in professional and enterprise editions, then each one of these requirements is an EAC. • If the requirements set responsible for authentication and authorization users cannot be implemented in home edition, but it can be implemented in professional edition, depending on customer´s choice, and it is implemented by default in enterprise edition, then each one of these requirements is an EAC.

Table 1 - Heuristics Description (cont.).

HEURISTICS FORM	
HIEAC-8	***Rule*:** If there are maintenance requirements that are neither part of the standard system solution nor of its editions, however these requirements need to be implemented in order to accomplish a customer´s customization, and the implementation of these requirements necessary add new attributes, variables, methods, relationships, classes and functionalities, then each of these maintenance requirements is an EAC.
	***Examples*:** • If a maintenance requirement, involving software improvement, request for implementing a new functionality to open doors automatically, and new attributes, methods needed to be declared and that functionality will not be incorporated into the standard solution, then it is an EAC. • If a maintenance requirement, involving adaptation, request to change GPS by GLONASS standard to capture latitude, longitude and altitude of a certain vehicle, and that adaptation will not be incorporated into the standard solution, then it is an EAC.
HIEA- 9	***Rule*:** If there are requirements which represent messages exchanges in object-oriented paradigm, involving systems, subsystems, modules, components and others, and these messages generate coupling, then each of these requirements is an EAC.
	***Examples*:** • If there are requirements which establish messages among Security, Persistence and Logging module, and this generate coupling, then each of these requirements is an EAC. • If there are requirements which establish messages among Accounting, Human Resources, Marketing and Selling modules, generating coupling, then each of these requirements is an EAC.
HIEAC-10	***Rule*:** If there are requirements of subsystems, packages, components, functionalities and others, under experimental or temporary conditions, or when are frequently changed, then each one of these requirements is an EAC.
	***Examples*:** • If a requirement which request for a tax implementation is a temporary functionality, then it is an EAC. • If autonomous navigation requirements set is a vehicle software module under testing, then that requirements set is an EAC.
HIEAC-11	***Rule*:** If there are requirements describing 24x7 functionality, or 24x7 functionalities that need to be changed, then that requirements set is an EAC. (Note: 24x7 functionality are functionalities that need to support operations 24 hours per day during 7 days per week).
	***Example*:** • If a requirement to log phone calls duration is needed to work 24 hours per day and 7 days per week, then it cannot stop and it is an EAC.
HIEAC-12	***Rule*:** If there are requirements triggered by other requirements, then each one of these requirements is an EAC.
	***Example*:** • If a requirement to count phones calls minutes is triggered after the requirement to complete the phones calls, then the requirement to count phones calls minutes is an EAC.
HIEAC-13	***Rule*:** If there are requirements which describe business rules, then each of them is an EAC.
	***Examples*:** • If a requirement “If a segmentation algorithm of images needs to be triggered by camera 1, then run algorithm 1, otherwise run algorithm 2” is a business rule, then it is an EAC. • If a requirement “Financial transactions over US$100.000,00 should be logged and information sent to Federal Police” is a business rule, then it is an EAC.
HIEAC-14	***Rule*:** If there are requirements that track the control-flow of a system or even identify the requirement that is responsible for trigger another one, then each of them is an EAC.
	***Examples*:** • The requirement “Always that the temperature of stove exceed 3200ºC, log every functionalities triggered by system operator” is a an EAC, because it needs to keep track about the established parameters. • The requirement “If the moving detection algorithm detects something using camera 1, then cameras 2, 3, 4, and 5 need record everything by 20 minutes” is an EAC, because it needs to identify the camera responsible for triggering it.

The EA Definition uses Specification of EA Candidate and Specification of Decision Equation artifacts as input. Such an activity enables an appropriate definition about which requirements are more qualified and present more advantages to be implemented as aspects, previously classified as EA Candidates by the earlier activity. The main focus of this activity is to assign scores to each of EA Candidates identified by the

previous activity. In the EA Definition activity, the higher the score assigned, the less will be the involved risks regards with the use of AO, improving chances to take competitive advantages using aspects. The execution of EA Definition activity consists of applying metrics and a Decision Equation, which are part of the Decision Equation Specification artifact, to each one of EA Candidates listed during Specification of EA Candidates. This way, the Specification of EA artifact is generated, as depicted in Figure 2.

The focal point of this paper is to present the first activity of the MEAID named EA Identification.

MEAID comprises a method, which guides the software engineer to use the Requirements Specification and Heuristics to Identify EA Candidates (HIEAC) artifacts as inputs to the Identification of EA Candidates activity. This generates the Specification of EA Candidates as an intermediate output to MEAID.

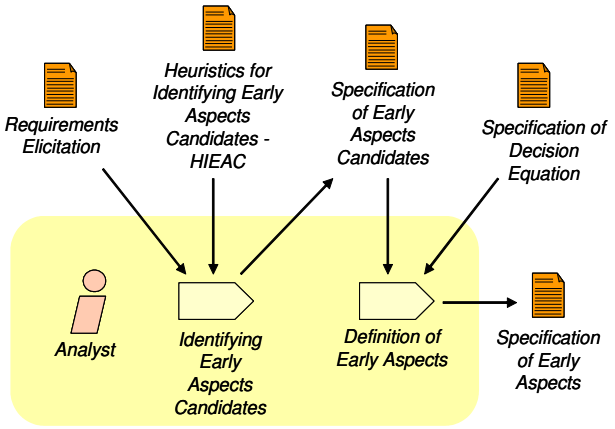

Figure 2 - MEAID´s Conceptual Modeling.

2.1 EA Candidates Identification

Aiming to support the Identification of EA Candidates, Table 1 shows up heuristics proposed on this work and inserted into an artifact named Heuristics to Identification of EA Candidates (HIEAC).

These heuristics enables the identification of EA Candidates when correctly applied into the Requirements Specification phase of a software system development, supporting software engineering professionals for reducing of empirical and subjective decisions.

The tabular notation, used in past to describe heuristics for Object Oriented Paradigm, has been adopted to describe MEAID´s heuristics. This way, the second column has been splitted into three rows to describe such heuristics by means of their rules and examples.

This notation was adopted from [11] and employed to describe methods related to OO, as illustrated in Table 2. The authors encouraged MEAID users to maintain the example and justification fields up-to-date, as each new application of the method is initiated.

Table 2 - Model to Describe Heuristics.

Abbreviation	**Rule:**
	Example(s):
	Justification(s):

The fields in Table 2 are:

- **Abbreviation** - represents a heuristic unique identifier;
- **Rule** - describes the practical rule to be applied by the software engineering in order to accomplish the expected results. For instance, this field can report the practical rule to so that it can aid to identify EA Candidates. Therefore, new EA can emerge during each application of this rule. In this case, software engineering shall carefully examine these new examples, so that they can be added to the Example field, aiming to be reused in future projects;
- **Example(s)** - report results discovered by earlier application of heuristics indicated by the Rule field. It enables the activity speed up to identify EA Candidates during the Aspect-Oriented Software Development (AOSD). Nevertheless, the Example field must not replace the Rule one, because the abstraction level of its description enables the appearance of new examples, due to particularities of each system;
- **Justification(s)** - exhibit fundamentals, examples or bibliographic references which justify the heuristic development as well as its application. Due to lack of space, this paper omits this field.

Despite heuristics justifications are based on [4] and [13], other documents were used to extract heuristics to indicate which requirements are potential enough to be implemented as aspects. Among these documents we can cite: programs developed by this paper´s first author, during the learning process as well as investigations about AO; scientific papers, such as [14]; technical books, such as [4], [15], [16], and [13]; and technical and user programming language manuals, such as [17], [18], and [19].

An additional effort has been made to select the best papers which give the theoretical foundations to each developed heuristic. Anyway, this comprises a hard

work, because the success of coding a requirement as an aspect depends on the technology used.

The authors of this paper are evaluating this method and believe that it can be necessary to break down the Justification into sub-items. This way it can enables to identify different technologies to program results of a heuristic using each of them. These can be based on the fact that these technologies are still in the development process, have distinct characteristics and AO are not yet well consolidated.

2.1 EA Definition

Aiming to define EA using EA candidates identified in previous activity, six technical factors and one organizational factor have been proposed, in order to reduce the risks involving implementation of requirements applying AO technologies.

The six technical factors are Documentation (Doc), Previous Experience (PE), Tangle Requirements (TR), Spread Requirements (SR), Interest in Components (IC) and Adjustment (Ad) and one organizational factor is AO Importance (AOI). Initially, these seven factors composed the criteria to understand why and when a requirement could be implemented applying AO technologies with low risks.

In order to calculate Adjustment value, It was developed the following equation:

$$Ad(Ri) = \frac{Adjusted\_Heuristics(Ri)}{Total\_of\_Heuristics} * 4$$

Equation 1 - Calculating Adjustment Factor.

The variables in Equation1 are:

- Ri → represents the requirement under analysis;
- Adjusted_Heuristics → Amount of heuristics that the requirement Ri satisfied;
- Sum_of_Heuristics → Current value is 14 (Table 1).

Likert Scales have been developed to support professionals on quantifying others Factors, as showed on figures below.

For Documentation factor, use the Likert Scale showed in Figure 3 to answer the following question: Is there a documented solution for requirement under analysis?

For Previous Experience factor, use the Likert Scale showed in Figure 4 to answer the following question: What is the previous experience of development team?

For Tangle Requirements, Spread Requirements, Interest in Components and AO Importance factors, use the Likert Scale showed in Figure 5 to answer the following questions, respectively: (Tangle Requirements) What is the tangling level of requirement under analysis? (Spread Requirements) What is the spreading level of requirement under analysis?

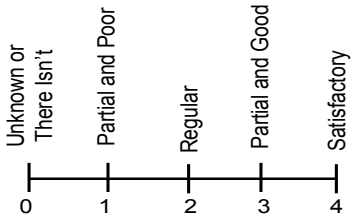

Figure 3 - Likert Scale for Quantify Factors.

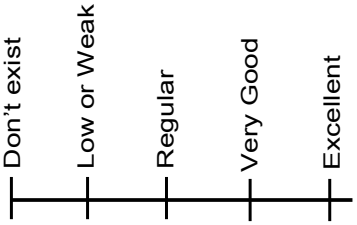

Figure 4 - Likert Scale for Quantify Factors.

(Interest in Components) What is the Interest level in transform the requirement under analysis in a component? (AO Importance) What is the strategic value of applying AO technologies to implement the requirement under analysis?

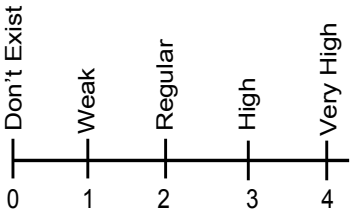

Figure 5 - Likert Scale for Quantify Factors.

Nowadays, several researchers, engineers, analysts, developers and so on are experimenting AO Technologies in your business all over the world. So, get answers for each question of all factors, considering each requirement of system under analysis, is very important to investigate and determine risks levels.

The Decision Equation, showed in Equation 2, was developed to support professionals on determining risk levels of each requirement and answer if the requirement could be implemented applying AO technologies.

To elaborate the Decision Equation, Technical and Organizational Factors were reclassified in Decision and Weight Factors. Decision Factors describe factors able to avoid use of AO Technologies if their value is zero. In

other case, the risks value increase or decrease according with directly of factors value.

The Decision Factors are Tangle Requirements, Spread Requirements and AO Importance. The Weight Factors are Documentation, Previous Experience, Interest in Components and Adjustment.

$$S(x) = 10 * \left[\frac{[Tr + Sr + OAI] * [\lceil \frac{Tr * Sr * OAI}{64} \rceil]}{12} \right] * \left[\frac{(\alpha * PE) + (\beta * Doc) + (\gamma * IC) + (\delta * Ad)}{16(\alpha + \gamma + \beta + \delta)} \right]$$

Equation 2 - Decision Equation.

As shown in Figure 6, nine curves are engaged to define the Decision Equation. Graphically Moderate Curve represents Decision Equation with exponent 1 (one). Curves named Bold, Bold+ and Bold++ represent Decision Equation with exponent 0.75, 0.55, 0.35, respectively. Curves named Conservative, Conservative+ and Conservative++ represent Decision Equation with exponent 1.4, 1.8, 2.2, respectively.

Curves named Superior Edge and Inferior Edge define thresholds that we need to respect to reduce risks. For example, if the result of Decision Equation, for some requirement, is under Inferior Edge, then the developer should not use OA Technology to implement it. If the result of Decision Equation, for some requirement, is above Superior Edge, then the developer should use OA Technology to implement it, because the risks are minimal.

However, there is a region in graphic considered Critic Region that is located between Superior and Inferior Edges. When the result of Decision Equation, for some requirement, is in Critic Region, then the developer should examine the value of each of Factors and determine whether is worth apply OA Technology to implement the requirement. As near the result equation is from Superior Edge, lower is risks and vice-versa.

Prototypes of AO System were used to define Curves values in order to research group determine weights, define what factors are decision factors or Weight Factors, propose Superior and Inferior Edges.

The Decision Equation has been submitted in sensitivity analysis and the expected behavior was achieved. It could have their factors and weights changed to represent, more correctly, the Enterprise Technology Use Politics, after adequate investigation.

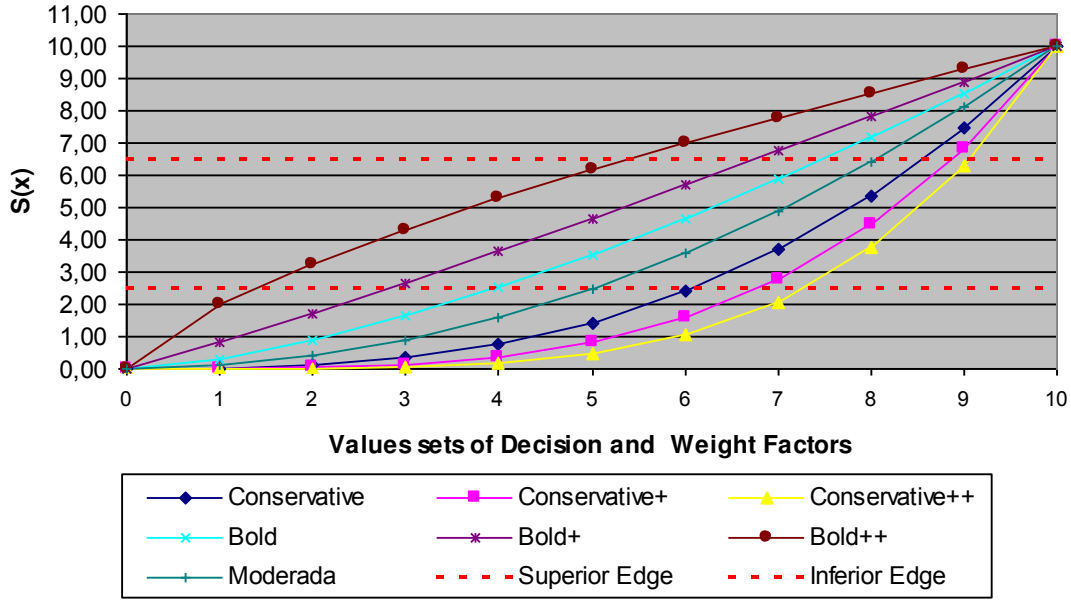

Figure 6 - Decision Equation and Variations3. Experimentation and Results.

The experimentation to validate MEAID has been applied at the Federal University of Lavras, involving 1 associate professor, 3 graduated and 26 undergraduate Computer Science students.

The first day was allocated to AO training, whereas the experimentation was applied in the second day. The collaborators were separated into 2 groups, called Control Group and Experimental Group. The Control Group has applied the Intuitive Method (IM), and it was used to compare efficacy and efficiency from the Experimental Group, which has applied MEAID. By comparing results of these groups, it was possible evaluate which group has achieved a better efficacy and efficiency during the Candidate Early Aspect Identification activity.

Table 3 presents a set of metrics created and used as parameters for comparing IM and MEAID.

Table 4 and Table 5 summarize results analysis involving the two methods applied. Both groups received the same AO training at the same time. IM´s appliers only have used knowledge learned during AO training and their intuition to classify requirements as Early Aspect Candidate, whereas MEAID´s appliers have been used to perform the same activity.

IM´s appliers have justified, into filled forms, why they believe that a requirement was an EAC. This way, Rough Results are the ones calculated taking into account answers without consider such a justification. The Refined Results were calculated by considering correct appliers´ justifications.

Table 4 and 5 summarize results obtained after applying Student's T-Test.

Table 3 - Metrics used to compare IM and MEAID.

TEAF - Total Early Aspects Found	**CEi** - Compared Efficiency
ATEAI - Average Time for Early Aspects Identifying	**Errors** - Errors Amount
CEa - Compared Efficacy	**ET** - Errors Tax

Table 4 - Summary of Analysis about Intuitive Methods Results (Rough Data) versus MEAID Result (Refined Data).

Metric	MEAID Diagnostic based on Averages	Intuitive Method (IM)	MEAID	T-Test (5%)
TEAF	2,78 times better	10,43	29	99,97
ATEAI	20% of time of IM	12,12	3,42	**
CEa	2,49 times more efficacy	28,19	61,7	99,79
CEi	2,64 times more efficient	5,79	15,34	99,91
Errors	3,29 times more amount	31,86	105	**
ETx	Performed ¼ of IM erros	56,89	13,39	100

** Collected data didn't fulfill statistical requirements for applying T Test

Considering the Decision Equation, superior and inferior edges had been defined after some Sensitivity Analysis application. Each factor assumed all values possible, considering 10% of variation. For example, all factors received value zero and the Documentation (Doc) factor received values 0, 0.1, 0.2, 0.3 until max value 4. Following, all factors received 0.1 and Documentation received values 0, 0.1, 0.2 until max value 4. All factors were frozen in some moments and modified in others moments, in order to test all possible arrangement.

Table 5 - Summary of Analysis about Intuitive Methods Results (Refined Data) versus MEAID Results (Refined Data).

Metric	MEAID Diagnostic based on Averages	Intuitive Method (IM)	MEAID	T-Test (5%)
TEAF	4,61 times better	6,29	29	99,99
ATEAI	13% of time of IM	20,63	3,42	**
CEa	3,63 times more efficacy	16,99	61,7	99,99
CEi	4,37 times more efficient	3,49	15,34	99,97
Errors	3,40 times more amount errors	30,86	105	**
ETx	Performed ¼ of IM erros	55,1	13,39	100

** Collected data didn't fulfill statistical requirements for applying T Test

Determine the best value to superior and inferior edges is a hard task yet, and new discussions are necessary to determine a better way or improvements in method.

4. Conclusion

This paper described the activities Early Aspects (EA) Candidates Identification and EA Definition comprised by Method for Early Aspect Identification, showing Heuristics and a Decision Equation developed to support it.

This paper represents a piece of results from an Doctoral Thesis and is based on "need of Scientific and Technological Software Engineering Community to have a systematic for identifying and defining of Early Aspects using Requirements Specification, in order to increase efficacy and efficiency of Aspect Oriented Software Development, reducing empirical and subjective decisions".

It has shown results from new method experimentation and its evidences to increase efficacy and efficiency of Early Aspects Identification activity, fulfilling an important gap of AO. MEAID provides reduction of empirical and subjective decisions, because Heuristics aid to identify reasons that lead software engineer to classify requirements as an Early Aspect Candidate.

Considering the Early Aspects Candidates Identification activity, Student's T-Test has been applied and showed evidences that MEAID is more effective and efficient than the Intuitive Method, with trustworthiness higher than 99.7%.

Considering the Early Aspects Definition activity, a Sensitivity Analysis has been applied, based on AO prototypes, for defining of references values able to support what requirement should be implemented applying AO Technologies.

The main advantages shown by MEAID were:

- Use of heuristics to identify EAC aiming to: a) transfer AO technology to organizations; and b) teach AO paradigm to students, teachers, researches and so on;
- Use Decision Equation to quantify the risks of apply OA Technology for implementing requirements;
- Reduction of empirical and subjective decisions by means of heuristics that facilitate to understand why a given requirement was classified as an EAC;
- Reduction of empirical and subjective decisions by means of Decision Equation that facilitate to understand why developer should use AO Technology for implementing requirements;
- Increase the amount of Early Aspect identification up to 4.61 times;
- Reduction in 83% of the average time for EAC Identification;
- Increase in the Efficacy and Efficiency of the Early Aspects Identification activity; and
- Reduction in Tax Errors - Etx;

The main disadvantages identified in MEAID include:

- Lack of assurance that all of the Early Aspects can be found; and
- Increase the chances of EAC be wrongly found. Anyway, this disadvantage is reduced or avoided by following an activity called Early Aspect Definition, not explained in this paper due to lack of space.

After 5 months analyzing experimentation results performed at Federal University of Lavras, there are intentions to apply MEAID in software industry. Considering that undergraduate students enrolled in first the period have applied MEAID correct and successfully, professionals of Software Engineering area might be apply it easier and collect even more expressive results. However, it is necessary to adapt and incorporate it into an organizational software development process before its application.

The heuristics set of the MEAID´s Early Aspects Candidate Identifying can be applied to help students learning AO paradigm as well as to find out Early Aspects faster and easier, similarly to OO heuristics.

It was hard define the values for superior and inferior edges, so it is very important consider new approaches, in order to allow comparisons among them and choose better solution.

References

[1] Ossher, H., Kaplan, M., Katz, A., Harrison, W., and Kruskal, V. (1992) "Specifying subject-oriented composition". TAPOS, 2(3):179-202. Special Issue on Subjectivity in OO Systems, 1992.

[2] Ossher, H. and Tarr, P. (1999) "Using subject-oriented programming to overcome common problems in object-oriented software development/evolution". In International Conference on Software Engineering, ICSE'99, ACM, 1999.

[3] Miller, S. K. (2001) "Aspect Oriented Programming Takes Aim at Software Complexity". Magazine IEEE's Computer, vol.34, no.4, pp.18-21, April, 2001.

[4] Resende, A. M. P. and Silva, C. C. (2005) "Programação Orientada a Aspectos em Java". Published by Brasport Livros e Multimídia Ltda, Rio de Janeiro, March 2005.

[5] Sardinha, A.; Chitchyan, R.; Weston, N.; Greenwood, P. and Rashid, A. EA-Analyzer: Automating Conflict Detection in Aspect-Oriented Requirements. In ASE '09: Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering . November 2009.

[6] Sampaio, A. and Rashid, A. Mining early aspects from requirements with ea-miner. In ICSE Companion '08: Companion of the 30th international conference on Software engineering. Leipzig, Germany. May 2008.

[7] Resende, A. M. P.; Silveira, F. F.; Cunha, A. M. (2005) Early Aspects: Some Analysis, Trends and Perspectives. Published in the Proceedings of the Early Aspects Workshop, held in conjunction with OOPSLA'05 - Object-Oriented Programming, Systems, Languages And Applications, San Diego, California, USA, October, 2005

[8] Rashid, A. and Chitchyan, R. Aspect-oriented requirements engineering: a roadmap. In EA '08: Proceedings of the 13th international workshop on Early Aspects. ACM. Leipzig, Germany. 2008

[9] Chitchyan, R.; Greenwood, P.; Sampaio, A.; Rashid, A.; Garcia, A.; Silva and Lyrene Fernandes. Semantic vs. syntactic compositions in aspect-oriented requirements engineering: an empirical study. In AOSD '09: Proceedings of the 8th ACM international conference on Aspect-oriented software development. March 2009.

[10] Zambrano, A.; Fabry, J.; Jacobson and G. Gordillo. S. Expressing aspectual interactions in requirements engineering: experiences in the slot machine domain, SAC '10: In Proceedings of the 2010 ACM Symposium on Applied Computing. Sierre, Switzerland. March 2010.

[11] Araújo, J. et al. Early aspects: the current landscape. Technical Report COMP-001-2005, Lancaster University, England, 2005.

[12] PUC-RIO (2005). Requirement Models by means of Business Process. 2005. Lesson Notes of subject called INF1364: Business and System Modeling, taught by Prof. Pedro Oscar de Souza Cruz. URL: <http://www.inf.puc-rio.br/~pedro/INF1364/ NotasAula-Modelo Requisitos.pdf>. Accessed: jan 07-05.

[13] Laddad, R. (2003) AspectJ in action: practical aspect oriented programming. New York: Manning Pub. 2003. 512p.

[14] Hannemann, J.; Kiczales, G. (2002) Design pattern implementation in java and AspectJ. Proceedings of the ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, 17th., 2002, Seattle. Proceedings…Seattle, USA: OOPSLA, 2002. Aspectwerkz. (2005) Plain Java AOP: Overview. Design and Development Site of ASPECTWERKZ, in URL: <http://aspectwerkz.codehaus.org/> accessed in: nov. 2005.

[15] Jacobson I.; Ng, Pan-wei. (2005) Aspect-oriented software development with use cases. Boston: Addison-Wesley, 2005.

[16] Filman, R. E.et al (2004). Aspect-oriented software development. Boston: Addison-Wesley, 2004. 800p.

[17] The AspectJ Project. (2003) AspectJ Project Site, Available in URL: <http://www.eclipse. org/aspectj/>. Access since: march, 2003.

[18] Jasco Publications. (2005) System and software engineering lab. Site of JAsCo System and Software Engineering Lab at:<http://ssel.vub.ac.be/jasco/publications>. Accessed in 06 october, 2005.

[19] Aspectwerkz. (2005) Plain Java AOP: Overview. Design and Development Site of ASPECTWERKZ, in URL:<http://aspectwerkz.codehaus.org/> accessed in: nov. 2005.