International Journal of Decision Support System Technology
Volume 14 • Issue 1

Towards Group Decision Support in the Software Maintenance Process

Dinedane Mohammed Zoheir, Université Oran 1, Algeria*
Abdi Mustapha Kamel, Université Oran 1, Algeria
https://orcid.org/0000-0001-8235-9209

ABSTRACT

Software maintenance is a key element of the life cycle of software. However, the techniques of software maintenance do not consider the diversity and the complexity of decisions which do not stop increasing. So, there are at present a few tools, susceptible to insure the relevance and the efficiency of the decision-making in this phase. The work presented in this paper aims to eliminate or at least to reduce the effect to fall in an expensive change by reducing the time to find a compromise on the adequate change. The development of the decision support system for software maintenance is an answer to the problem. The developed tool allows one to make a fast diagnosis on the software by using the coupling metrics; to help the decision-makers of the maintenance, according to their preferences often conflicting; to adopt a change among several proposed. To answer this group decision where various points of view are considered, the authors propose a negotiation protocol. This protocol tries to find a compromise that suits best all the decision makers.

KEYWORDS

Coupling Metrics, Decision Support, Negotiation Protocol, Software Maintenance

INTRODUCTION

The maintenance is the last phase of the software life cycle, it is the most expensive phase. According to (Pfleeger & Bohner, 1990), Software maintenance is defined as the process of modifying the software system or components, to correct faults, to improve its performance, or to adapt to changes in the environment (Dhillon, 2002). The maintenance cost depends on the dependency degree between the entities of software architecture. A change can have considerable and unexpected effects on the rest of the system. The danger incurred during a modification is the change propagation. Therefore, it's better to have an idea of the software architecture to estimate the change impact and so reduce the maintenance cost. The modularity is considered as an important criterion of the software quality. A software product is said modular if its components present a weak degree of coupling. Within the framework of Object Oriented applications, there are various types of coupling between classes. Measuring these types of relations allows understanding better the link which exists between the coupling of the classes and the quality attributes (Aggarwal et al., 2006). We define a change in a program as a modification brought to one of its elements (class, method or variable). So, the impact is seen in our context as the consequence of a change. The impact analysis is an activity where the objective is to determine the extent of a change request. It estimates affected elements at the level of

DOI: 10.4018/IJDSST.286677 *Corresponding Author

the source code. The more a class is coupled with other classes, the more it is sensitive to the changes made in these classes and the more it may undergo errors.

Several studies have been made in this direction. (Abdi et al., 2009), used a probabilistic approach, other (Cheikhi, 2005), worked on predicting the impact of change through machine learning, (Ren et al., 2004), developed a tool for assisting developers with program understanding and debugging. The common point is the use of coupling metrics as an indicator of impact analysis.

However, In the case of a decision making in maintenance process, coupling property is not the only indicator of maintenance, various activities such as understanding, readability and modification of programs are considered to be a factor in reducing the estimate of maintenance effort. The skills of software maintainers involve knowledge of the programming language. This factor includes the experience and familiarity of the maintainers with the latter so that the maintenance team can easily understand the system (de Souza, Anquetil, & de Oliveira, 2005) (Anquetil & de Oliveira, 2005).

In software maintenance, the managers are confronted to a problem of the group decision where several often conflicting points of view should be considered.

In the face of various proposed changes, the choice of the most relevant change is a complex problem, especially when we are in a group environment. In this context of research, the decision-making support in software maintenance is motivated by the lack of tools. Offering a tool which allows helping the managers in the choice of the change among several represent a financial gain to companies. The proposed approach allows the development of a decision support system collective which tries to bring an effective and rational solution. We exploit in our approach the advantages of the Multi-Agent Systems (Tučník, 2010), to model the decision-makers, so their interactions to arrive at a beneficial consensus by basing itself on a negotiation protocol.

PROBLEM AND CONTRIBUTION

With the evolution of the software systems, an important stream of changes must be taken in charge as well as their propagation in the rest of the system. Successfully changing the software of a disciplined way while maintaining its functioning and its integrity with a reasonable cost requires a change impact analysis before its implementation. Indeed, the team of maintenance has to be able to supply answers to the following questions: what type of change is it? What is the extent of the change? What are the criteria of the engineer?

In the management of the software projects, several changes can be proposed to solve the same problem at the level of code and satisfy the same need for the user of a software system. However, different objectives must be considered when it is about a group of decision-makers. For that purpose, we propose an approach of group decision making support which allows supplying the best assistant to facilitate the choice of a change, and so to optimize the cost. Minimizing the cost, is to minimize the time between the proposal of the change, its implementation, and its realization. Choose the right change in the right moment while considering the differences between the members of the group is a difficult task. This work aims to help a group of decision-makers in the choice of appropriate change by taking into account the interests of each decision maker to reach an agreement. In this context of research, the decision support system uses the advantages of the Multi-Agent Systems to represent the diversity of the actors implied in the decision. We endow the MAS module with a negotiation protocol based on mediation. This protocol includes an initiator agent responsible for the negotiation process and a set of participant agents. The agents represent the different actors of the management team of the maintenance involved in the decision. The final choice of change in this situation will be made after a process of negotiation. Section 2 reviews the works concerning the impact analysis, decision support systems and the reaching agreements in MAS. The third section is reserved for our approach. Section 4 explains the experimentation and implementation of the tool. The perspectives of this work are cited in the conclusion.

RELATED WORK

Coupling vs Impact Analysis

The coupling measures the strength of the interconnection between the modules of a system. During the process of software maintenance (Chidamber & Kemerer, 1994), the coupling predicts the difficulty of change of the program modules and their implications in the programs in the other modules (Kabaili, 2003). (Lounis & Melo, 1997), the coupling refers to the degree of interdependence between classes. More a class is coupled with other classes, more is important its sensibility for the changes in these classes. Good quality software has to obey the principle of weak coupling (Chidamber & Kemerer, 1994, Dahane et al., 2019). A weak coupling facilitates the maintenance seen that dependencies between the classes are small. For this reason, we have introduced in our tool a module which makes it possible to calculate the coupling metrics and exploited them as factors which influence maintenance.

Decision Support Systems (DSS)

Group Decision Support Systems (GDSSs) are a special class of group work support systems intended to provide computational assistance to collaborative decision-making processes. GDSSs define a widely used collaborative technology that has increased user participation and the quality of decision-making. Using a GDSS, group members can discuss a problem, exchange ideas, evaluate alternative solutions, vote and choose a solution. GDSS is supposed to improve the group's performance by improving the decision-making process.

A lot of contributions regarding decision-making support were published. We can quote in a not exhaustive way the works of (Taghezout & Zaraté, 2009), which propose an architecture based DSS in order to solve some uncertainly problems, (Adla et al., 2011), propose a toolkit for GDSS facilitators to increase the ability to monitor and control the GDSS process, (Yazdani et al., 2017), propose a decision support model for selecting logistics providers, the research provides a platform to facilitate decision process; (Recio-García et al., 2013), which used social factors in the argumentation and the negotiation for the DSS of group; (Fan & Shen, 2011), shows the effect of the use of decision-making support of group in the management studies of the value; (Wibowo & Deng, 2013), present an approach based on the consensus to solve effectively the problem of decision-making of group multi-criterion; (Shen et al., 2012), propose an approach to integration of systems with low coupling for the decision support regarding management and regarding maintenance; (Hamdadou & Libourel, 2011), propose a decision support system of for the industrial diagnosis by using the MAS and the multicriteria analysis, and finally (Oufella & Hamdadou, 2018), propose a group negotiation model based on argumentation.

Reaching Agreements in MAS

Multi-Agent Systems have proved their effectiveness for modeling but one of the major problems encountered is to reach agreement on a particular problem. The agent sends messages according to his preferences to find an acceptable consensus. But agents face a dilemma: on the one hand, they want to maximize their own preferences, and on the other hand, they have to negotiate to reach a compromise. In MAS, several techniques are used to reach agreements, mainly auctions (Verrons, 2004), voting systems (Sandholm & Lesser, 1996), negotiation (Grant et al., 2000), and argumentation (Hamdadou, 2008, Oufella & Hamdadou, 2018). In MAS, a negotiation is a form of interaction that allows a group of agents to reach an agreement. In general, there are different negotiations protocols in MAS applications, the most used are: monotonous concession protocol, one-step protocol, contractual protocols (the net contract protocol, the extended net protocol of the contract) (Verrons, 2004).

Proposed Approach

Our approach consists of two main modules: the module of coupling metrics extraction and the MAS module. The identification and the understanding of the changes which can be brought to the

objects oriented applications turn out important and fruitful. For that purpose, we have inspired by the work realized in (Cheikhi, 2005). We took back a questionnaire to collect the perceptions of people which ensure the maintenance of the software know the nature of the changes. The changes are grouped by class, method, and attribute, and the result of this questionnaire is either the change is often, rare, or never.

We extract the concerned coupling metrics of all the classes from a source code Java and each engineer proposes a change to solve a problem at the code. The final result will be an arrangement of these changes from the best at least good according to the preferences of the decision-makers.

Let us note that it is possible that a change whose nature is "often" is applied to a strongly coupled class. As it can have a change whose nature is "rare" is applied to weakly coupled class. Furthermore, for every change proposed by an engineer, we have indications of the criteria of evaluation (experience in the domain, familiarization with the environment of the system, experience in the programming language, and the cost of this change). So we take into consideration all these criteria to try to find a change that satisfies the group according to their preferences.

The performance matrix in our case includes in lines the proposed changes or the solutions for the resolution of a problem (the actions); in columns, the various coupling metrics of the classes carrying the change, the nature of change, and the criteria of every engineer who proposed the change (the criteria). For example, if solution1 (change of signature of a method of the classe4), the criteria will be the various metrics of this classe4, the nature of this change, and the criteria of the engineer who proposed the change. The decision-makers are then invited to introduce their preferences. These data are going to show at the end an arrangement of changes for every decision-maker according to their preferences, and then the MAS supplies the process of negotiation to reach a final agreement that satisfies the largest number of decision-makers.

The Metrics Extraction Module

The purpose of the metrics extraction module is to capture important features like coupling. We chose the coupling property for two reasons:

- To have an idea on the quality of the system in terms of coupling, and this allows to estimate the change propagation in the system.
- To exploit these metrics as criteria in the performance matrix.

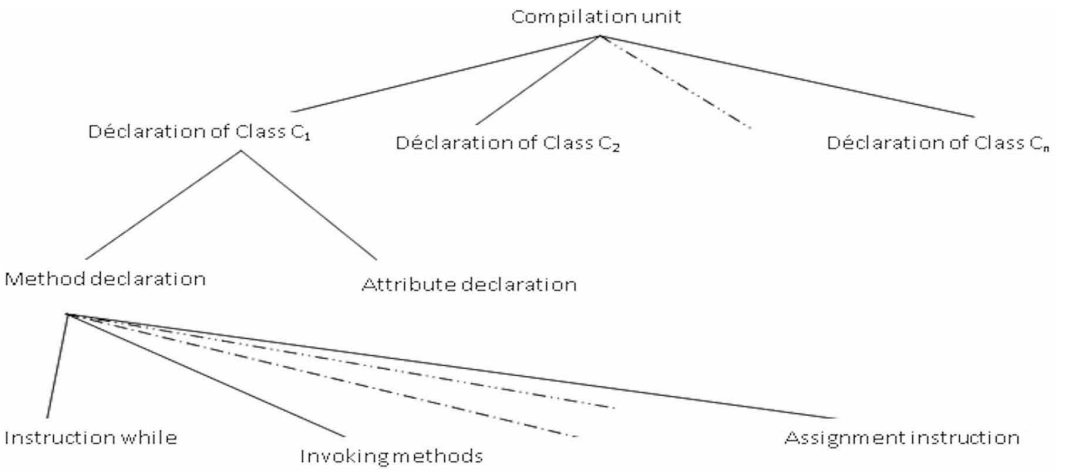
The authors develop a tool under Eclipse which allows to analyze a source code Java and to extract all the important data for the calculation of the considered coupling metrics for every class of the system. This analysis is insured by plugin ASTParser. The metrics of every class carrying out the change are exploited as criteria in the performance matrix.

For a problem at software written in Java, or a new need for the user of the latter, several solutions (changes at the source code level) are proposed to answer the same need, the impact analysis allows to choose the adequate solution, and the least expensive, by considering several aspects: the nature of change, its impact on the rest of the system, and the quality of the engineer. In our work, the authors attribute indices to every change, for the change never (index 0), the rare change (index 1), and the change often (index 2).

Source Code Analysis Process

This process allows us to analyze a source file Java and to supply its syntactic tree (AST) which corresponds to a set of nodes representing all the constituents of the system until instructions as shown in Figure 1. It is a tree structure of objects.

Figure 1. Abstract Syntactic Tree (AST)



In our work, we try to browse the project tree, file by file, analyze it and bring out the information useful for the implementation of metrics, namely: classes, superclasses, descendants, attributes, methods, function calls, etc. For this, we used the notion of Visitor: "Visitor".

The visitor represents "an operation to be performed on the elements of an object structure.Visitor lets you define a new operation without changing the classes of the elements on which it operates." (Gamma, 1995). This term generally means Analysis. It refers to classes of objects that visit other objects during the browse and perform appropriate tasks.

ASTVisitor is an abstract class associated with a pair of "visit (T node)" and "endVisit (T node)" methods for the different categories of nodes in the syntax tree; these are access to an array, assignment, instruction blocks, break and continue statements, for, while loops, etc.

Visit (T node): allows you to visit the AST node and perform certain operations. If the returned value is True, then the child nodes will be visited. This class does not provide any implementation, subclasses must be defined, and the latter implement the method as needed. For example: visit (CompilationUnit node) allows you to visit the Java source file, visit (MethodInvocation node) allows you to visit function calls and visit (IfStatement node) allows you to visit if statements.

EndVisit (T node): visits the node to perform operations. This method is called after all the child nodes have been visited. The default implementation of this class does no task. Subclasses must be defined to relocate the method.

Description of the Implemented Metrics

The metrics chosen in the context of the study are presented in Table1.

The authors consider it useful to present the definition of certain metrics that we have implemented.

CBO (Coupling Between Object):

$$CBO(c) = \mid \{d \in C - \{c\} \mid uses(c,d) \cup uses(d,c)\} \mid$$

Represents the number of classes with which a class is paired. Two classes are said to be coupled, if the methods defined in one use methods or instance variables of the other. A class c is coupled to a class d if it uses d or is used by d. The definition of the "uses" predicate (c, d) requires that the

Table 1. List of implemented metrics

DAC	Data Abstraction Coupling: This metric represents the number of non-inherited attributes that are of type one of the classes of the application.
CBO	Coupling Between Object: Represents the number of classes with which a class is coupled.
RFC	Response For a Class: Represents the number of methods invoked in response to a message.
MPC	Message Passing Coupling: Counts only method invocations from other classes.
AMMIC	Ancestors Method -Method Import Coupling: Corresponds to the number of parent classes with which a class has a method-method interaction and an IC type coupling.
OMMIC	Others Method -Method Import Coupling: Corresponds to the number of classes (other than superclasses and subclasses) with which a class has a method-method type interaction and an IC type coupling.
DMMEC	Descendants Method - Method Export Coupling: Corresponds to the number of subclasses with which a class has a method-method interaction and an EC type coupling.
OMMEC	Others Method - Method Export Coupling: Corresponds to the number of classes (other than superclasses and subclasses) with which a class has a method-method type interaction and an EC type coupling.
CBO'	Coupling Between Object: Counts the number of classes with which a class c is coupled and not having an inheritance relationship.
CBO-IUB	CBO Is Used By: This metric consists of the number of classes using the target class
CBO-U	CBO Using: Represents the part of the CBO that is interested in the classes used by the target class c

method which uses the class d and the method used or the attribute of the class d be implemented in their classes.

CBO' (Coupling Between Object):

$$CBO'(c) = \mid \{d \in C - (\{c\} \cup Ancestors(C)) \mid uses(c,d) \cup uses(d,c)\} \mid$$

Counts the number of classes with which a class c is coupled and not having an inheritance relation. This definition does not take into account all the ancestors of a class.

RFC (Response For a Class):

$$RFC(c) = \{M\} \cup \{\underset{alli}{R_i}\}$$

{M} is the set of methods of the class.

{Ri} the set of methods called by method i.

Represents the number of methods invoked in response to a message. It is worth the number of methods of the class and the number of methods called by each method m of the class c.

MPC (Message Passing Coupling):

$$MPC(c) = \sum_{m \in Mi(c)} \sum\nolimits_{m' \in SIM(m) - Mi(c)} NSI(m, m')$$

Represents the number of messages sent by one class to other classes. MPC (c) counts only the method invocations of other classes and not the specific method invocations to class c.

MAS Module

This module aims to represent the various actors who have their objectives, and their preferences of decision. Multi-agent technology has already shown its ability in different domains. Especially, in the applications of group decision-making support thanks to the ease which it supplies.

The authors delegate to the MAS the selection of the change elected according to the negotiation process. To cope with this group decision, it is necessary to pass by a negotiation procedure to reach a beneficial consensus. To achieve this, the authors endow the MAS with a negotiation protocol base on the mediation involving two types of agents:

- The initiator agent (manager): it is the agent responsible for the management of the negotiation, the modification of the contract and the choice of the final elected change.
- The participant agents (group): are the agents involved in the decision; the objective of every agent is that its favorite change is chosen.

It is essential that the participating agents pass by a phase of negotiation, according to a well-structured protocol, to reach a beneficial agreement to the group.

The negotiation is made between the initiator agent and all the participating agents. Once the model to be built is described, the authors need the infrastructure to implement it. They use JADE (Java Agent Development framework) (Bellifemine et al., 2002), to generate agents and to implement the process of negotiation between them. Finally, the tool provides a consensual solution which is a compromise. In the following section, the authors will describe JADE, and after they will detail the negotiation process between agents.

JADE Platform

JADE: JADE (Java Agent Development framework) is a multi-agent platform created by the TILAB laboratory described by (Bellifemine et al., 2005, Bellifemine et al., 2008). JADE allows the development of multi-agent systems and applications conforming to FIPA standards (Fipa, 2002). It is implemented in JAVA and provides classes that implement "JESS" for the definition of agent behavior. JADE has three main modules (necessary for FIPA standards).

1. DF "Director Facilitor" provides a "yellow pages" service to the platform;
2. ACC "Agent Communication Channel" manages communication between agents;
3. AMS "Agent Management System" supervises the registration of agents, their authentication, access and use of the system.

These three modules are activated each time the platform is started.

There are tools that have been developed in the JADE platform in order to support the difficult task of debugging multi-agent applications. Each tool is packaged as an agent, obeying the same rules, the same communication possibilities and the same life cycles of a generic agent.

- RMA Remote Management Agent: The RMA allows you to control the life cycle of the platform and all the agents that make it up. Several RMA can be launched on the same platform as long as they have different names.
- Dammy Agent: The DammyAgent tool allows users to interact with JADE agents in a special way. The interface allows the composition and sending of ACL messages and maintains a list of sent and received ACL messages.
- Agent Directory Facilitator: The DF interface can be launched from the RMA menu. This action is in fact implemented by sending an Agent Communication Language ACL message to the DF asking it to load its graphical interface.
- Agent Sniffer: When a user decides to spy on an agent or a group of agents, it uses a sniffer agent. Each message leaving or going to this group is picked up and displayed on the sniffer's interface.
- Agent Inspector: This agent is used to manage and control the life cycle of a running agent and the queue of its sent and received messages.

The Phases of Protocol Negotiation

The current negotiation protocol is based on the Contract-Net Protocol (Davis & Smith, 1983). It is characterized by a series of messages exchanged between the initiator agent and the participant agents. It proceeds in three phases: the proposal phase, the conversation phase, and the decision phase.

1- The proposal phase: this phase is the first phase of our protocol, it introduces the negotiation, includes the proposal of the contract by the initiator to the participants and collects of the answers of each of the participants. Every participant can either accept or refuse the proposal.
2- The conversation phase: this phase intervenes only when the participants were not enough to accept the proposal of the contract of the initiator and when the counterproposals are authorized. The initiator indicates then to the participants this possibility. A conversation between the initiator and the participants takes place during which proposals of modifications are exchanged. Further to the modifications proposed by the participants, the initiator proposes a new contract. So, a new phase of the proposal begins.
3- The final decision phase: This final decision phase results in either the confirmation of the contract or the cancellation of the contract. This decision is made by the initiator according to the participants' responses.

Characteristics of the Proposed Protocol

During the negotiation process, many fundamental aspects must be taken into account, such as:

- The language used by the agents to exchange information during the negotiation (primitives and strategies);
- The objects of negotiation;
- The strategies adopted by the different agents;
- The cardinality of the negotiation.

In what follows, we present the main characteristics of the proposed negotiation protocol.

The Objects of Negotiation

Resources are the objects of negotiation; they can be either personal or shared. In our case, they are common resources (the proposed changes).

The Cardinality of Negotiation

The complexity is an important characteristic for the negotiation. It is one of the main reasons why it is necessary to use negotiators agents. Let us examine the complexity in the number of messages inferred by the protocol. In the worst case, the number of messages can be O (mn) where n is the depth of the process and m the number of agents implied in a negotiation.

The Primitives of Negotiation

To lead a negotiation process to its term, it is necessary to define primitives for the initiator and other specific primitives for the participants.

Initiator Primitives:

- Propose (contract): it is the first primitive that the initiator sends to the participants in order to propose a contract to them. The contract contains different resources to negotiate.
- Modification request (contract): this message indicates to the participants that the contract cannot be concluded in its current form and that it must be modified.
- Confirm (contract): this message indicates to the participants that the contract is confirmed. The negotiation is successful.
- Cancel (contract): this message indicates to the participants that the contract is canceled. The negotiation is failed.

Participants Primitives:

- Accept (parameters): this message responds to the offer of the contract made by the initiator. The participant indicates through this message to the initiator that he accepts the contract as it is.
- Refuse: this message responds to the offer of the contract made by the initiator. The participant indicates to the initiator that he refuses the contract.
- Propose modification: this message responds to a modification request from the initiator. The participant sends the initiator a list of possible changes to the contract (a counter-proposal). The number of changes contained in the list is a parameter of the negotiation.

The Agent's Strategies

The authors resumed the protocol which distinguishes two roles: initiator and participant. The strategy of negotiation is not the same according to the role of the agent, there are thus two sorts of strategies.

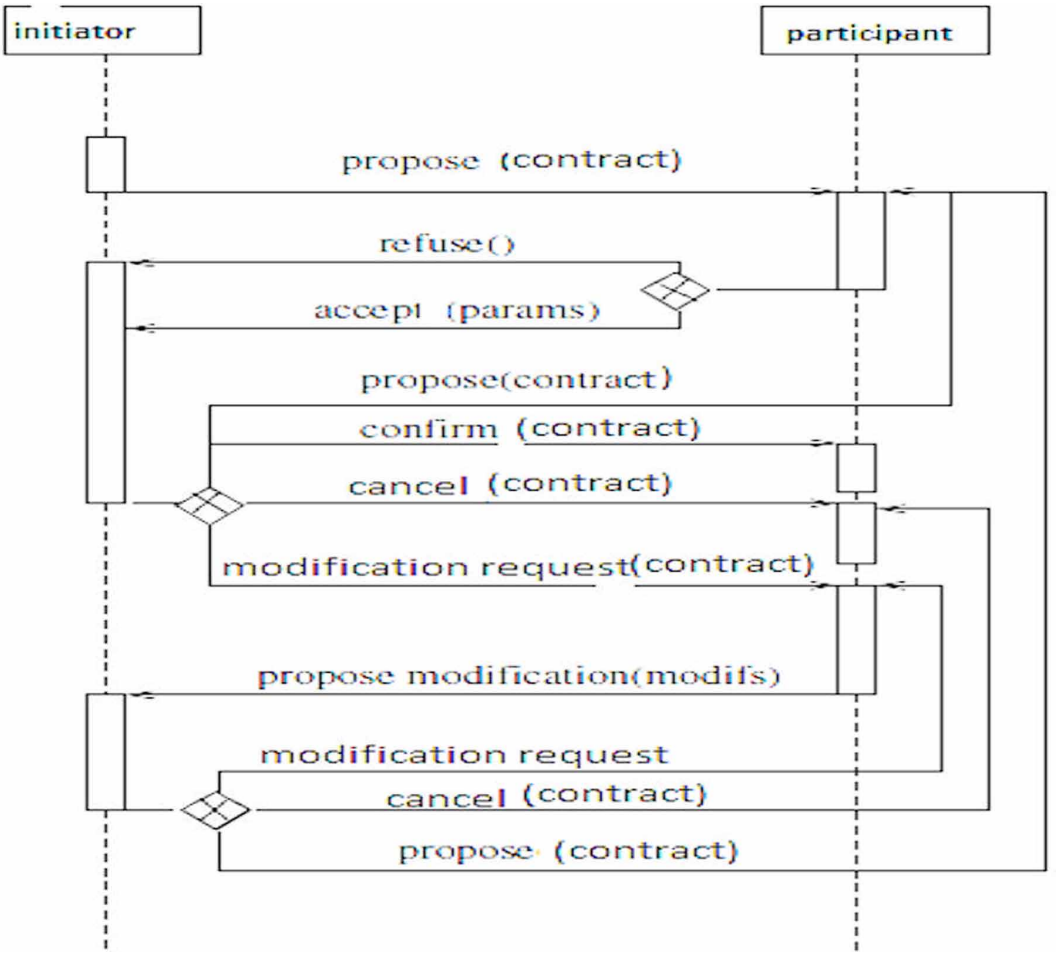
The strategy of the initiator allows him to decide to confirm or to cancel the contract and to synthesize the various proposals of modification of the participants in order to propose a new contract. Another strategy is one of the participants who uses it to decide to accept or to refuse the proposal of contract and to find a modification for the contract when the initiator in demand.

Participant Strategies: we associate with each participant agent three strategies:

The strategy of establishing preferences: each participant assigns priorities to the different resources according to its preferences. The priority is always an integer between 1 and 10; each agent is free to change the priorities it has given at any time. Participant Agent compares its preferences with the criteria of each change to assign a priority to each resource. For example, if for a change, all the preferences of the participant are satisfied then 10 is awarded. Otherwise, if 70% are satisfied then a priority is given from 7 to 9, and so on using a random function.

The strategy of acceptance: the negotiation can proceed in several rounds until a compromise is found. In each new round, the participant receives a new proposal. If it corresponds to its preferences

Figure 2. Graph of interaction between an initiator and a participant



at the round t, it accepts this proposal. Else, it checks whether the proposition corresponds to one of an earlier preferences. If it is the case, it accepts the contract indicating its actual preferences.

The strategy of refusal: when the participant does not receive a proposition that corresponds to its preferences at round t, nor other earlier preferences, it refuses it and makes against the proposal which corresponds to its preferences at round t.

Initiator Strategy: we associate with the initiator only one strategy used at the time of the modification phase. Before throwing the negotiation, the initiator attributes priorities to the participating agents according to their profiles. The priority is an integer from 1 to 10.

The strategy of modification: when the participants are not rather numerous to accept the initiator's proposal, the latter is obliged to modify its contract for the next round while taking as a starting point all modifications sent by the participants at round t, in order to find a new possibility for the contract.

For that, the initiator associates a score SCORE (R_i) with each resource R_i(i= 1,.. n) which takes into account the priority of the participant agent as well as the relative priority of the resource for the participant. As in the method of scorages (Ferber, 1995), the resource which has obtained the highest

score at round t will be the winner resource and the initiator will propose it in the new contract. This score is updated each time the participants have been less numerous to accept the contract. The SCORE (R_i) is given by the following equation:

$$SCORE(Ri) = \text{priority}(\text{Ri}, \text{init}) * \text{priority}(\text{init}, \text{init}) + \sum_{j=1}^{m} \text{priority}(\text{Ri}, \text{partj}) * \text{priority}(\text{partj}, \text{init}) \cdot$$

- n, m: the number of resources and decision-makers respectively;
- priority (R_i, init) is the priority of the resource R_i for the initiator. For reasons of equity between the initiator and the participants, we take into account the same number of resources for the initiator as for the participants.
- priority (init, init) is the priority which the initiator gave itself.
- priority (R_i, $part_j$) is the relative priority of the resource laughed for the participant $part_j$.

Indeed, the participant does not communicate the priorities which it gives to these resources, the initiator considers while the first resource sent is the most priority at the participant's and so attributes it the maximal priority. Then it decrements the priority of 1 for the following resource, without coming down under the minimal value. If the resource is not a part of the received list, the authors attribute it a discriminating priority equal in 0.

EXPERIMENTATION

The implementation of the proposed group decision support process is accompanied by developing a tool. The goal is to support the approach proposed by a tool and real data. Besides, testing on a real case allows validation of our approach.

For experimentation, the authors used several systems of small and average sizes. But to validate the experiment, BOAP system is used (Alikacem & Snoussi, 2002): it is an application developed with the Java language, supplied by the CRIM Laboratory. BOAP (Toolkit for the Analysis of Programs), is a set of integrated software tools who allows an expert to estimate quickly the quality level of software. The authors chose BOAP because it has already used as test system in others works (Abdi, 2007, Abdi et al., 2009), to verify certain hypotheses about change impact analysis and prediction. The characteristics of BOAP are presented in the following table:

Implementation of the Tool

The extraction option (Figure 3) allows browsing the source code of the system in entry and giving its AST.

After obtaining the AST, the authors are going to proceed to the structuring in classes, methods, attributes, inheritance, method invocation,...etc. The structure option (Figure 4) allows posting this structuring of the source code under an exploitable form to facilitate the manipulation.

The metrics calculation (figure 5) is made based of information extracted from the Figure 4.

- Definition of Actions: The actions are changes proposed by different engineers to meet a need in the source code. They need to be negotiated to help decision-makers choose the change that will be retained. The decision is taken by considering a set of criteria and taking into account the different points of view of the different actors involved in the decision. The changes proposed for our experimentation are:

Table 2. Characteristics of the test system (BOAP)

Characteristics	BOAP System
Number of files	424
Number of modules	22
Number of independent classes	103
Number of classes	208
Number of basic classes	117
Number of methods	107513
Number of attributes	1401

Figure 3. Extraction of the AST

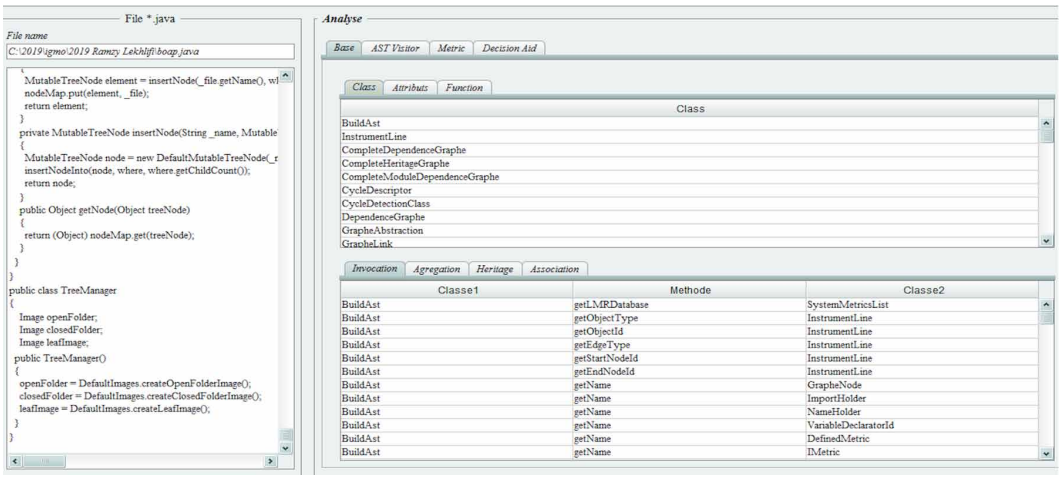


Figure 4. Source code structure

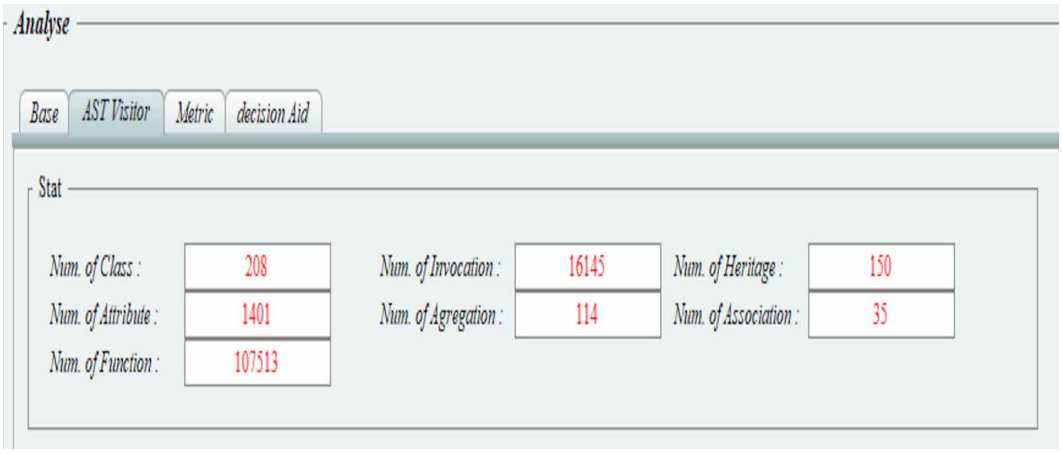
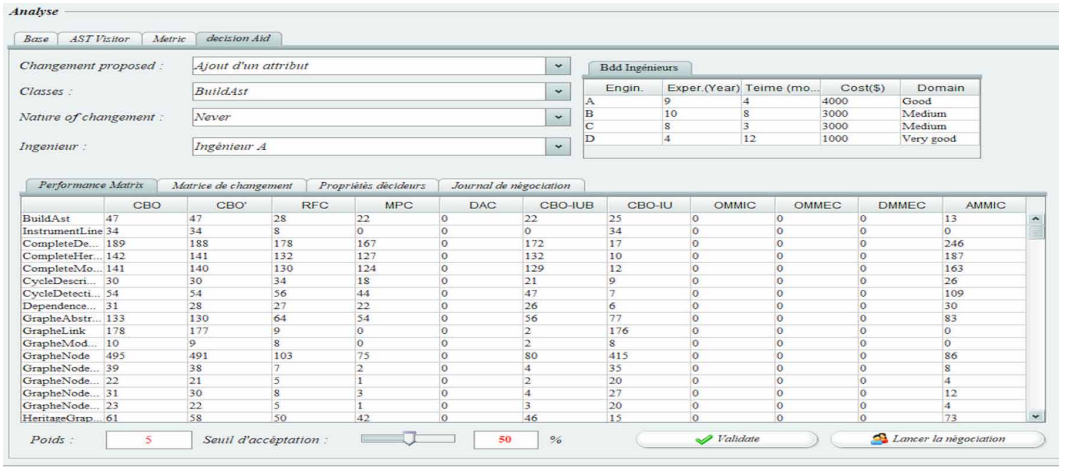


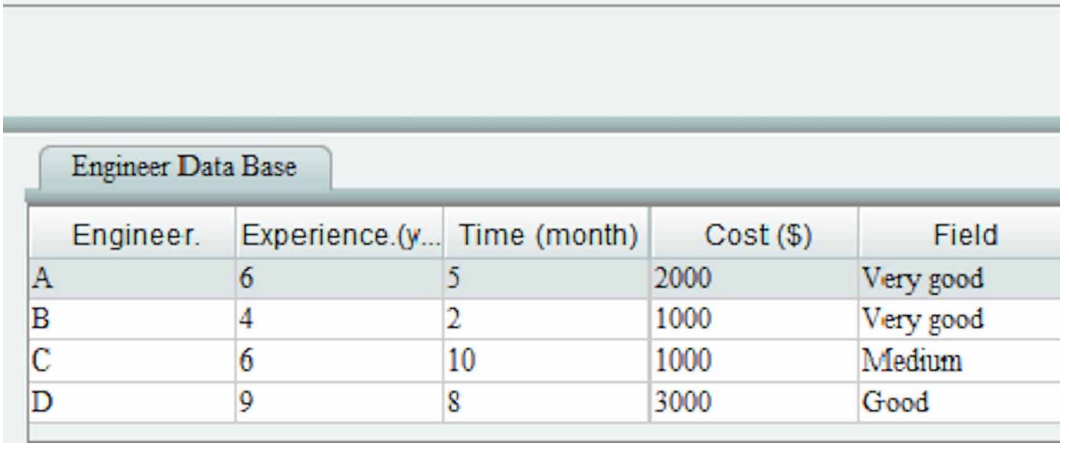
Figure 5. Calculation of considered metrics



Change1: Addition of a method to BuildAst class.
Change2: Addition of a superclass to DependancePanel class.
Change3: Change of an attribute type of GraphModuleLink class.
Change4: Change of a method type of CompleteModule class.
Change5: Change of implementation of a method of laModuleDependance class.

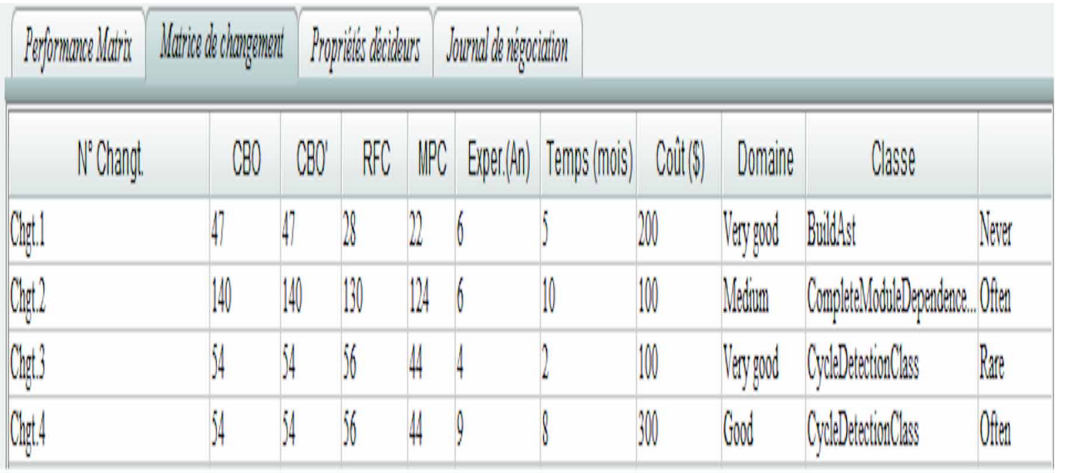
Figure 6 shows the attribution to each change the criteria of the engineer who proposes it (experience, delay, estimated cost, knowledge in the field).

Figure 6. Database of the engineers criteria



After the proposition of the changes by the engineers, the matrix of proposed changes is obtain, and illustrated in Figure 7.

Figure 7. Proposed changes Matrix



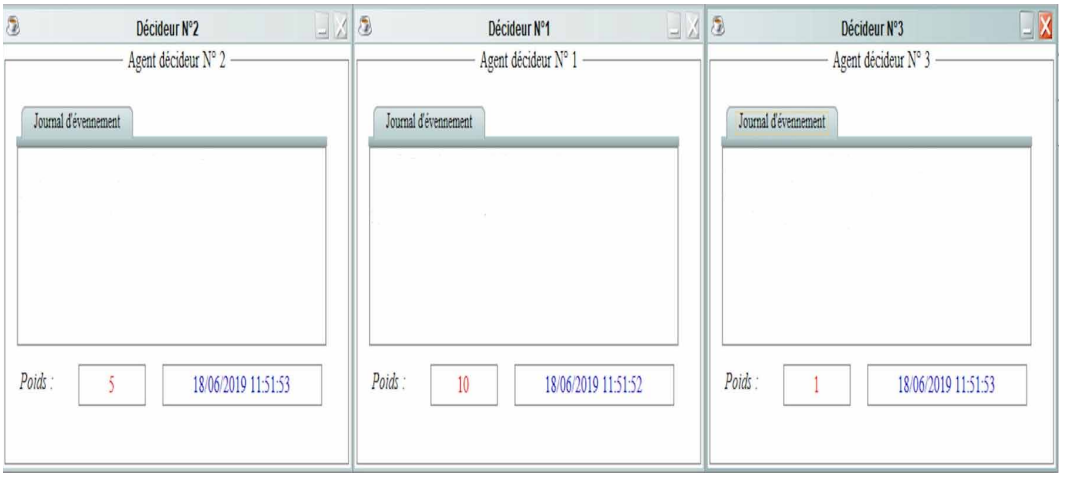
N° Changt.	CBO	CBO'	RFC	MPC	Exper.(An)	Temps (mois)	Coût ($)	Domaine	Classe	
Chgt.1	47	47	28	22	6	5	200	Very good	BuildAst	Never
Chgt.2	140	140	130	124	6	10	100	Medium	CompleteModuleDependence...	Often
Chgt.3	54	54	56	44	4	2	100	Very good	CycleDetectionClass	Rare
Chgt.4	54	54	56	44	9	8	300	Good	CycleDetectionClass	Often

-Identification of Decision Makers: in our case, the group decision-makers are:
DM1: Project Manager, **DM2:** Analyst, **DM3:** Chief Programmer, **DM4:** Finance Responsible.

We note that in MAS modeling, the DM1 is the initiator, and other DMs are the participants.

The initiator attributes to each decision maker a priority expressing its weight in the negotiation according to their profile as shows the Figure 8.

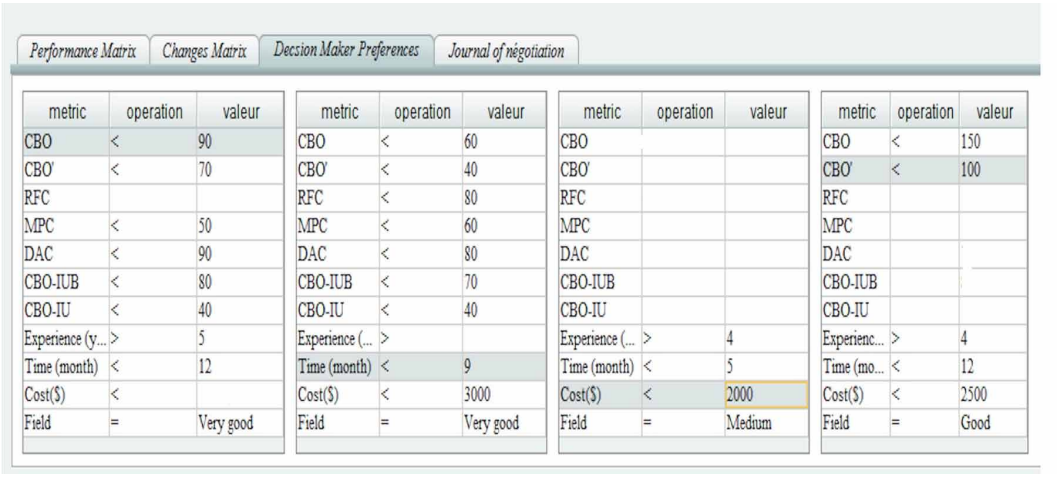
Figure 8. Allocation of priorities to Decision-Makers



The decision-makers are invited to introduce their preferences as shows the following Figure 9.

After the application of a random function, obtaining vectors of change for every decision-maker according to their own preferences (the priorities of each change for each decision-maker) as mentioned in figure 10.

Figure 9. Decision-Makers preferences



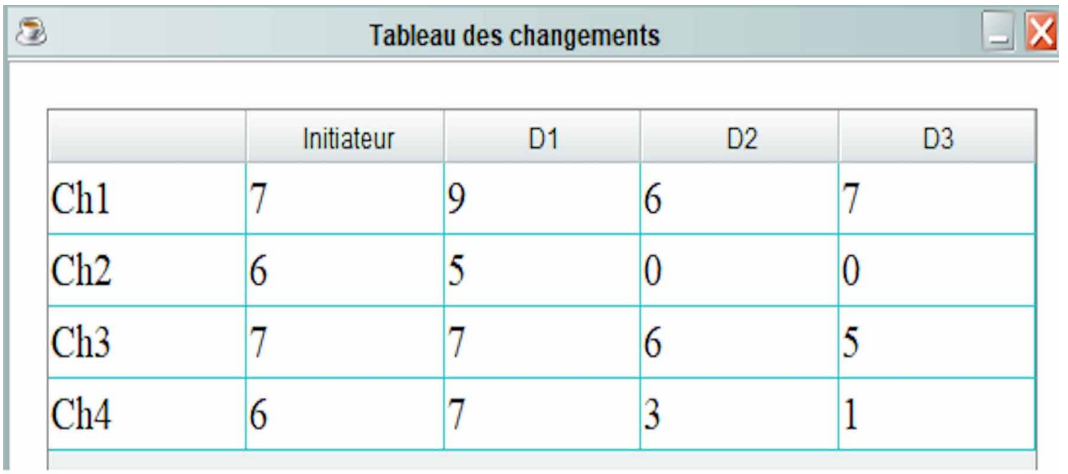
metric	operation	valeur
CBO	<	90
CBO'	<	70
RFC		
MPC	<	50
DAC	<	90
CBO-IUB	<	80
CBO-IU	<	40
Experience (y...	>	5
Time (month)	<	12
Cost($)	<	
Field	=	Very good

metric	operation	valeur
CBO	<	60
CBO'	<	40
RFC	<	80
MPC	<	60
DAC	<	80
CBO-IUB	<	70
CBO-IU	<	40
Experience (...	>	
Time (month)	<	9
Cost($)	<	3000
Field	=	Very good

metric	operation	valeur
CBO		
CBO'		
RFC		
MPC		
DAC		
CBO-IUB		
CBO-IU		
Experience (...	>	4
Time (month)	<	5
Cost($)	<	2000
Field	=	Medium

metric	operation	valeur
CBO	<	150
CBO'	<	100
RFC		
MPC		
DAC		
CBO-IUB		
CBO-IU		
Experienc...	>	4
Time (mo...	<	12
Cost($)	<	2500
Field	=	Good

Figure 10. Priorities of changes for each DM



	Initiateur	D1	D2	D3
Ch1	7	9	6	7
Ch2	6	5	0	0
Ch3	7	7	6	5
Ch4	6	7	3	1

Before beginning the negotiation process, it is necessary to fix a threshold of acceptance (70% of decision makers are satisfied); the minimal number of agreements necessary to conclude the contract (3 decision-makers), and the maximal number of rounds of negotiation (10 rounds).

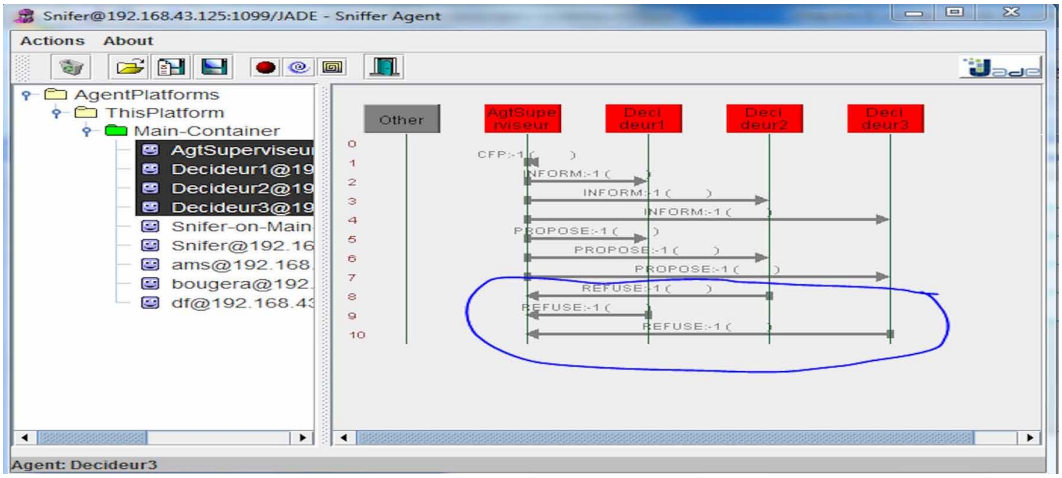
In this example, five changes are considered: Ch1 to Ch5 and the initiator proposes a three-party contract: D1, D2, and D3. The initiator has the maximum priority (10), a priority equal to 10 to D1, a priority of 5 to D2, and the priority1 to D3.

Note that D1 will send the changes in the following order Ch5; Ch4, Ch3,

Ch2, Ch1
D2 will send the resources in order: Ch1, Ch3, Ch4, Ch2, Ch5
D3 will send resources in order: Ch2, Ch4, Ch3, Ch1, Ch5

Suppose each participant sends only one resource per change request.

Figure 11. Refusal of the contract



D1 then sends the change Ch5, D2 sends Ch3, D3 sends Ch2 and for the initiator the change Ch2. After the first modification request, the notes will be:

- note (Ch1) = 0 * 10 + 0 * 10 + 10 * 5 + 0 * 1 = 50
- note (Ch2) = 10 * 10 + 0 _ * 10 + 0 * 5 + 0 * 1 = **100**
- note (Ch3) = 0 * 10 + 0 * 10 + 0 * 5 + 0 * 1 = 0
- note (Ch4) = 0 * 10 + 0 * 10 + 0 * 5 + 0 * 1 = 0
- note (Ch5) = 0 * 10 + 10 * 10 + 0 * 5 + 0 * 1 = 100

The initiator will therefore propose a new contract for the Ch2 resource. Suppose this contract is rejected again by the participants (which is consistent since none of them proposed this change).

The initiator will again request a modification to the participants. D1 and D3 will then propose Ch4, D2 will propose Ch3 and the initiator Ch3. The notes will then be updated as follows:

- note (Ch1) = 50
- note (Ch2) = 100
- note (Ch3) = 9 * 10 + 9 * 5 = **130**
- note (Ch4) = 0 + 9 * 10 + 9 * 1 = 99
- note (Ch5) = 0

The ranking of the changes then becomes: Ch3, Ch2, Ch4, Ch1, Ch5. Since Ch2 has already been proposed, it no longer counts for the selection of new changes to propose. This ensures that the proposals will stop at the latest when all possible solutions have been proposed.

This time, the initiator will propose a contract for the Ch3 resource to the participants. This proposal is likely to succeed, based on the priorities assigned to Ch3 by the participants.

- note (Ch1) = 50 + 8 * 10 = 130
- note (Ch2) = 100
- note (Ch3) = 130 + 8 * 10 + 8 * 1 = **218**
- note (Ch4) = 99 + 8 * 5 = 139

- note (Ch5) = 0

After several modifications of the contract and in the third round, the decision-makers arrive at a consensus; the selected change is the change 3.

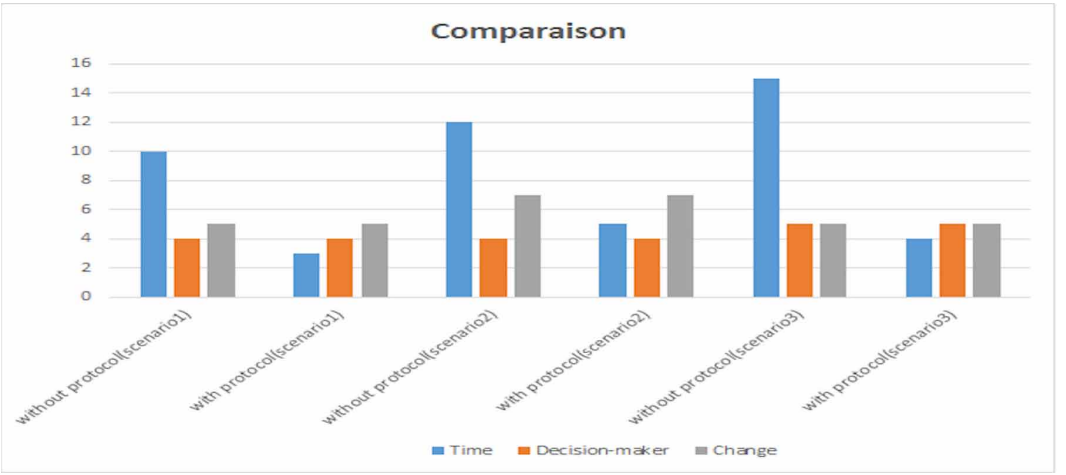
According to Figure 9, the change3 is ranked as the second priority for the initiator, third for the decision maker1 who also has a significant weight, the first for decision maker 2, and the third for the decision maker3, which explains that he had a negotiation that was made to try to find a compromise between the decision-makers in order to satisfy as many decision-maker as possible.

Comparative Analysis

In this section, the authors empirically assess the negotiation framework for the group of decision-makers. The experiments were done with three different scenarios: scenario1, the case with four decision makers and five changes. The second scenario, the number of proposed changes is increased to seven, and the number of decision makers is kept at four. The third scenario, the authors left the number of changes at five, and increased the number of decision makers to five. In all three scenarios, the aim is to show the value of using the tool in minimizing the time taken to reach consensus among decision-makers.

The experiments are designed to test the hypothesis that the use of the negotiation protocol is useful for reaching consensus among group members quickly. Furthermore, our expectation is that the improvement achieved through negotiation will increase as the number of agents involved in negotiation increases. Three bars are displayed: Time, Decision-Makers and changes for the three scenarios as mentioned in figure 12.

Figure 12. Comparative study of the negotiation



After the launch of the experiment, the authors notice that the first scenario shows that using the negotiation activates the convergence towards a compromise between the decision makers. In addition, the second scenario also shows that the group of decision-makers quickly came to a compromise when they used the protocol (5 seconds with the protocol versus 12 seconds without). In the third scenario, we clearly see the effectiveness of using the protocol. Comparing with the first scenario, the time to reach a compromise without protocol is 15 seconds (10 seconds for the 1st scenario) while with the protocol the duration is 4 seconds (3 seconds for the 1st scenario).

Following these experiments, the authors showed that either by increasing the number of changes (scenario1 vs scenario2), or by increasing the number of decision-makers (scenario1 vs scenario3), the use of the developed tool is effective insofar as it reduces convergence towards a consensus.

DISCUSSION

The change selection problem in software maintenance is usually a complex problem that involves a wide range of possible alternatives (changes) and multiple evaluation criteria. Most of the time, these alternatives and conflicting criteria are often evaluated by a number of decision-makers affected by the selection of the best possible change. From this perspective, having a group decision support tool or system is more than desirable in order to make the decision that is most appreciated by the group, despite the conflicting interests and preferences of the members of the group involved in the decision making.

Several studies have been made in the field of software maintenance, (Ruhe, 2002, Dingsøyr et al., 2009), describes fundamental principles and expectations. (Félix et al., 2010), propose a tool designed to assist software maintainers to solve software maintenance problems. (Kamaludeen et al., 2013), used the concept of expert system to record the knowledge of the information system in a knowledge base and inference logic to analyze the recorded knowledge. (Bouslama, 2012), proposed a knowledge-based approach to analyze and estimate quality factors in object systems. The author used Weka software to knowledge machine learning and build predictive models. The results obtained in the context of various experiments allow us to predict and estimate certain quality factors such as maintainability, reusability and predisposition to faults.

As interest in involving multiple stakeholders participation in evaluating decision problems grew, the need to incorporate the different decision makers affected by a group decision became an increasing focus to research efforts. However, the majority of proposed studies intended overlook this detail.

Group decisions are usually more complex compared to individual decision making, as several factors are involved, such as: conflicting individual goals, personal opinions, goals. To facilitate decision-making between the group to reach an agreement, negotiation seems the most appropriate solution.

To support this idea and facilitate decision making, we propose a decision support system for software maintenance reinforced by three areas of great importance: coupling metrics as an indicator of the impact of change, MAS to model the behavior of decision-makers, and negotiation through a protocol to find consensus among decision-makers.

The proposed system makes it possible to: analyze a source code and calculate certain coupling metrics, integrate the participation of the different MDs in a decision-making process, taking into account their perceptions, and find a consensus alternative (change) that satisfies a group of decision makers through a negotiation protocol.

Although this approach has given satisfactory results, there is still the case where the negotiation protocol arrives at round 10 without finding a consensus, we believe remedied this by adding a voting system this improvement will be taken into consideration in the perspectives.

CONCLUSION

The relevance of every change at the level of the source code during the software maintenance differs from a decision-maker to another one. Every proposed change has advantages and disadvantages of implementation and realization. Since there are usually several suggested solutions, the decision-makers regarding maintenance are often confronted with difficult choices. Their decisions should ensure the relevance of the change chosen according to the objectives often contradictory to be achieved by each of them. The objective of this paper is to propose a group decision-making support process

for facilitating the choice of a change through the integration of a negotiation protocol implemented in a MAS.

So, in this paper the authors have focused on proposing an approach that allows to:

- Have an idea of the quality of the system by the calculation of certain coupling metrics.
- Show the representation of the multiplicity of the decision-makers in the maintenance, their diversity, their behavior and their interaction to select the most appropriate change.

An application on a real system was proposed and served as a basis to demonstrate the feasibility of such approach. To strengthen the contribution, a negotiation protocol is developed to allow finding a beneficial consensus between the various actors of decision taking into account the preferences of every decision-maker. The authors cite some research perspectives as extension of our approach by considering other types of metrics and developing a negotiation protocol based on the argumentation.

REFERENCES

Abdi, M. (2007). *Analyse et Prédiction d'impact de Changement dans un système à objets.* Thèse de Doctorat d'Etat en Informatique, Université d'Oran, Es-Sénia.

Abdi, M., Lounis, H., & Sahraoui, H. (2009). *Predicting change impact in object-oriented applications with bayesian networks.* Paper presented at the 2009 33rd Annual IEEE International Computer Software and Applications Conference. doi:10.1109/COMPSAC.2009.38

Adla, A., Zarate, P., & Soubie, J.-L. (2011). A proposal of toolkit for GDSS facilitators. *Group Decision and Negotiation*, *20*(1), 57–77. doi:10.1007/s10726-010-9204-8

Aggarwal, K., Singh, Y., Kaur, A., & Malhotra, R. (2006). Empirical Study of Object-Oriented Metrics. *Journal of Object Technology*, *5*(8), 149–173. doi:10.5381/jot.2006.5.8.a5

Alikacem, E., & Snoussi, H. (2002). *BOAP 1.1. 0, Manuel d'utilisation*. CRIM, Janvier.

Anquetil, S. d. S. N., & de Oliveira, K. (2005). *A Study of the Documentation Essential to Software Maintenance.* Academic Press.

Bellifemine, F., Bergenti, F., Caire, G., & Poggi, A. (2005). *JADE—a java agent development framework. In Multi-agent programming*. Springer.

Bellifemine, F., Caire, G., Poggi, A., & Rimassa, G. (2008). JADE: A software framework for developing multi-agent applications. Lessons learned. *Information and Software Technology*, *50*(1-2), 10–21. doi:10.1016/j.infsof.2007.10.008

Bellifemine, F., Caire, G., Trucco, T., & Rimassa, G. (2002). Jade programmer's guide. *Jade Version, 3*, 13-39.

Bouslama, R. (2012). *Vers un système d'aide à la décision pour la conception en génie logiciel: une approche basée sur les connaissances*. Academic Press.

Cheikhi, L. (2005). *Estimation de l'impact du changement dans les programmes à objets*. Academic Press.

Chidamber, S. R., & Kemerer, C. F. (1994). A metrics suite for object oriented design. *IEEE Transactions on Software Engineering*, *20*(6), 476–493. doi:10.1109/32.295895

Dahane, M., Abdi, M. K., Bouneffa, M., Ahmad, A., & Basson, H. (2019). Using Design of Experiments to Analyze Open Source Software Metrics for Change Impact Estimation. *International Journal of Open Source Software and Processes*, *10*(1), 16–33. doi:10.4018/IJOSSP.2019010102

Davis, R., & Smith, R. G. (1983). Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, *20*(1), 63–109. doi:10.1016/0004-3702(83)90015-2

Dhillon, B. S. (2002). *Engineering maintenance: a modern approach*. CRC Press.

Dingsøyr, T., Bjørnson, F. O., & Shull, F. (2009). What do we know about knowledge management? Practical implications for software engineering. *IEEE Software*, *26*(3), 100–103. doi:10.1109/MS.2009.82

Fan, S., & Shen, Q. (2011). The effect of using group decision support systems in value management studies: An experimental study in Hong Kong. *International Journal of Project Management*, *29*(1), 13–25. doi:10.1016/j.ijproman.2010.01.008

Félix, Y., April, A., & Desharnais, J.-M. (2010). Software maintenance decision support system. *Proceedings of the 11th International Conference on Product Focused Software*. doi:10.1145/1961258.1961290

Ferber, J. (1995). Les Systèmes multi-agents: Vers une intelligence collective, InterEditions, 1995. *Connaissances et compétences requises Cette thèse s' adresse à des étudiants ayant un profil Intelligence Artificielle/Base de Données et elle nécessite d'avoir des compétences théoriques dans les domaines, 1.*

Fipa, A. (2002). *Fipa acl message structure specification.* Foundation for Intelligent Physical Agents. http://www. fipa. org/specs/fipa00061/SC00061G. html

Gamma, E. (1995). *Design patterns: elements of reusable object-oriented software*. Pearson Education India.

Grant, J., Kraus, S., & Perlis, D. (2000). A logic for characterizing multiple bounded agents. *Autonomous Agents and Multi-Agent Systems*, *3*(4), 351–387. doi:10.1023/A:1010050603219

Hamdadou, D. (2008). *Un Modèle d'Aide à la Décision en Aménagement du Territoire, une Approche Multicritère et une Approche de Négociation* (PhD thesis). Université d'Oran, Algérie.

Hamdadou, D., & Libourel, T. (2011). A MultiCriteria Group Decision Support System for Industrial Diagnosis. *INFOCOMP. Journal of Computational Science*, *10*(3), 12–24.

Kabaili, H. (2003). *Changeabilite des logiciels orientes objet: Proprietes architecturales et indicateurs de qualite*. Academic Press.

Kamaludeen, R. A., Cheah, Y.-N., & Sulaiman, S. (2013). *Software maintenance expert base decision support (soxdes) framework*. Paper presented at the 2013 International Conference on Advanced Computer Science Applications and Technologies. doi:10.1109/ACSAT.2013.13

Lounis, H., & Melo, W. (1997). *Identifying and measuring coupling in modular systems*. Paper presented at the 8th International Conference on Software Technology ICST'97.

Oufella, S., & Hamdadou, D. (2018). A collaborative spatial decision support system applied to site selection problems. *International Journal of Applied Management Science*, *10*(2), 127–156. doi:10.1504/IJAMS.2018.092078

Pfleeger, S. L., & Bohner, S. A. (1990). A framework for software maintenance metrics. *Proceedings. Conference on Software Maintenance 1990*. doi:10.1109/ICSM.1990.131381

Recio-García, J. A., Quijano, L., & Díaz-Agudo, B. (2013). Including social factors in an argumentative model for group decision support systems. *Decision Support Systems*, *56*, 48–55. doi:10.1016/j.dss.2013.05.007

Ren, X., Shah, F., Tip, F., Ryder, B. G., & Chesley, O. (2004). *Chianti: a tool for change impact analysis of java programs*. Paper presented at the ACM Sigplan Notices. doi:10.1145/1028976.1029012

Ruhe, G. (2002). *Software engineering decision support–a new paradigm for learning software organizations*. Paper presented at the International Workshop on Learning Software Organizations.

Sandholm, T. W., & Lesser, V. R. (1996). *Negotiation among self-interested computationally limited agents*. Citeseer.

Shen, W., Hao, Q., & Xue, Y. (2012). A loosely coupled system integration approach for decision support in facility management and maintenance. *Automation in Construction*, *25*, 41–48. doi:10.1016/j.autcon.2012.04.003

Taghezout, N., & Zaraté, P. (2009). Supporting a multicriterion decision making and multi-agent negotiation in manufacturing systems. *Intelligent Decision Technologies*, *3*(3), 139–155. doi:10.3233/IDT-2009-0061

Tučník, P. (2010). Multicriterial Decision Making in Multiagent Systems–Limitations and Advantages of State Representation of Behavior. *Advances in Data Networks, Communications, Computers,* 105-110.

Verrons, M.-H. (2004). *GeNCA: un modèle général de négociation de contrats entre agents*. Academic Press.

Wibowo, S., & Deng, H. (2013). Consensus-based decision support for multicriteria group decision making. *Computers & Industrial Engineering*, *66*(4), 625–633. doi:10.1016/j.cie.2013.09.015

Yazdani, M., Zarate, P., Coulibaly, A., & Zavadskas, E. K. (2017). A group decision making support system in logistics and supply chain management. *Expert Systems with Applications*, *88*, 376–392. doi:10.1016/j.eswa.2017.07.014

Dinedane Mohammed Zoheir is currently a Ph.D. student.

Abdi Mustapha Kamel holds a Master's degree and a Ph.D. degree in computer science from Department of Computer Science at the University of Oran 1 Ahmed Ben Bella, Algeria. He is currently, professor for the same Department. His research interests include the application of artificial intelligence techniques to software engineering, software quality, formal specification, Systems analysis and simulations, Data-Mining and Information Research.