ADAPTIVE GUIDANCE MODEL BASED SIMILARITY FOR SOFTWARE PROCESS DEVELOPMENT

Hamid Khemissa, Mohamed Ahmed-nacer ,Abdelkader Belkhir

Computer Systems Laboratory, Computer Science Institute, USTHB University,
El Alia BP n°32, Bab Ezzouar Algeria. Tel/Fax (00)213 21247917

ABSTRACT

This paper describes a modeling approach SAGM (Similarity for Adaptive Guidance Model) that provides adaptive and recursive guidance for software process development. This approach, in accordance to developer needs, allows specific tailored guidance regarding the profile of developers. A profile is partially or completely defined from a model of developers, through their roles, their qualifications, and through the relationships between the context of the current activity and the model of the defined activities. This approach aims to define the generic profile of development context and a similarity measure that evaluates the similarities between the profiles created from the model of developers and those of the development team involved in the execution of a software process. This is to identify the profiles classification and to deduce the appropriate type of assistance to developers (that can be corrective, constructive or specific).

KEYWORDS

Software process modeling, Process engineering, Adaptive guidance profile, similarity measure.

1. INTRODUCTION

Improving quality and productivity of software development requires assisting developers at both methodology level and consistency results level [1]. A guidance model in software engineering should combine the needed features to build the support system [2, 3].

Several PSEEs (**Process**-Centered **Software Engineering Environments**) [2, 4] deal the assistance aspect in the support of the software product development. Some PSEEs use an assistance description structured in steps like prescribing systems or proactive systems to control the operations carried out by the developer. The main limitations of these PSEEs are:

- The human actor has a central role in the progress of the development process regardless of his profile (qualifications and behavior).
- The basic guidance is defined as a global orientation core whatever the profiles of both the activity and the developer.
- The selection of the appropriate type of guidance is often more intuitive and not suitable.

To respond to these limits, several studies [2, 3, 5, 6] try to offer more flexibility in the language of software process modeling and a more adapted base of support and control. This tendency aims to define interventions of direct and adaptive assistance during the software process progress [7]. The following PSEEs included in the M1 level are as:

DOI : 10.5121/ijsea.2014.5205

ADELE/APEL is based on a reactive database. It proposes a global assistance of proscriptive type and automates part of the development process using triggers [8, 9].

RHODES/PBOOL+ uses an explicit description of a development process. The software processes are modeled in PBOOL language [10]. The activities are associated to a guidance system with various scenarios of possible realization.

ADDD/ALADYN provides process automation and control the impact in a concrete system. The task hierarchy is used to organize the process descriptions, called policies. Several aspects are grouped and treated in a policy. A policy can be instantiated for several tasks. The instantiated triggers are rules of the form event-condition-action (ECA) and used to implement a reactive behavior [11].

On the M2 level of Meta model, SPEM [12] introduced the concept of "Guidance" in the "*Managed Content*" package by defining the stereotype "Guidance". According to SPEM," the Guidance is a describable element which provides additional information to define the describable elements of a modeling. It also offers, through the stereotype "Guidance_kind" different types of guidance such as: Template, Guidelines, Checklists, etc. ..

However, the selection of guidance types remains defined in a manual and in an intuitive way. It depends on the experience and on the informal personality of the project manager. In addition, the proposed guidance is not adaptive to the actor's profile (role, qualifications and behavior).

In considering the principal limitations of PSEEs and essential characteristics of our approach as the context adaptation aspect and the abstraction levels, a comparative table of the studied Meta models is as follows:

Table 1. A Comparative table of the studied Meta models

Meta model / Criteria	ADELE /APEL	RHODES / PBOOL+	ADDD / ALADYN	SPEM
Global guidance core	Global	Global	Customized for each task	Global
Human performer profile oriented	Not adapted	considered strategy Model	Not adapted	Not adapted
Context development Guidance	Not adapted	Adapted	Adapted	Not adapted
Guidance types	Not invoked	associated with a specific guide	Not invoked	Intuitive selection
Explicit activity abstraction	Explicit abstraction	Implicit abstraction	Implicit abstraction	Explicit abstraction
Explicit task abstraction	Implicit abstraction	Not invoked	Explicit abstraction	Explicit abstraction
Process Modeling Language(PML)	APEL With predefined primitives	PBOOL+ With explicit primitives	ALADYN Not explicitly mentioned	UML Profile With explicit

The current tendency is that developers would like to have integrated environments that are suitable to specific needs according to the role and the characteristics of each developer and closed to the context of the underway task. However, the provided efforts to develop such environments remain an insufficient contribution.

In this context, our conceptual model is based on the conventional reasoning of software processes enriched by the "Adaptive Guidance" element which supervises the running of the activities. It also provides adaptive support to the actor. It is described by the following figure:

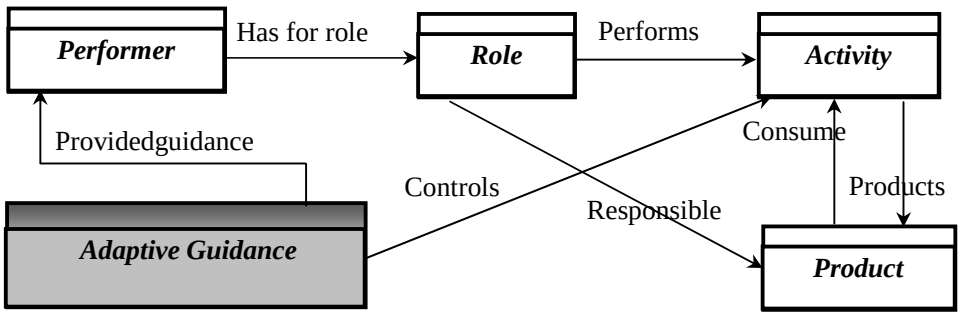

Figure 1. : Conceptual Model with Adaptive Guidance

This tendency of adaptive guidance environments is yet a subject of much research focusing on defining the concepts and objectives of process modeling software-based adaptive guidance [2, 3, 6].

For the sake of productivity and development time, our goal is to establish an optimal relationship between profile of guidance type and the adapted developer's profile to the context of development. The context is defined by the activity model, the developer and team development [2, 3, 13]. It is interesting to have an operator to assess the similarities within the handled data; this operator is the similarity function [13, 14, 15, 16]. The numerical similarity measures turn out to be extremely flexible employment. They are able to work on a broad spectrum of data types and it is fairly easy to introduce into the calculation, (statistical approximations if the underlying information is complex). In addition, similarities quantification by a continuous value implies that it is always possible and easy to compare pairs of objects. This is not the case for the symbolic similarity; treatment of numerical values is often done in an unsatisfactory way by rewriting these values in symbolic form [14, 16].

Our approach operates in the optimization of profiles classes in relation to the semantics of data manipulation. It defines a system for processing the similarity index and classification of guidance profiles. For this, we have to design a classifier in order to facilitate the analysis of our population and the type of assistance offered to appropriate developers involved in a software process.

The second section summarizes the technique of assessing similarity and the profile concept. Section three presents our approach (to model similarity with the software process), in addition to the implementation and practical evaluation of our approach by giving algorithms and related results. The last section concludes and presents future works perspectives.

2. SIMILARITY MEASURE AND PROFILE CONCEPT

Our approach operates by the similarity measure in the optimization of profile classes applied to the profile concept of each element identified in the software process.

2.1 Similarity measure

A similarity measure is defined on the set N (developers, documents, websites ...). Each object is described by m features. Each feature can be present or absent in every object. A measure of similarity, denoted by s, (between the elements of N is a specific application of NxN in R and satisfying some properties [14, 15]).

Examples of the use of similarity techniques are described in cases of heterogeneous binary data [17]. To transform a direct measure of similarity s into a dissimilarity measure d, we can apply the following formula: **$d(x, y) = smax – s(x, y)$.**

Thus, each element x is associated to a binary vector (x_1, x_2, **…**, x_m) such that:

$$x_i = \begin{cases} \textbf{1} & \textbf{If the feature i is present in the object x} \\ \textbf{0} & \textbf{Else} \end{cases} \qquad \textbf{For } i \in \{1,2,\ldots m\}.$$

The m characteristics are considered of equal importance and each object has at least one feature present. Note by:

- **a: The number of common characteristics between x and y.**
- **b: The number of features present in x but not y.**
- **c: The number of features present in y but not x.**
- **d: The number of missing features in x and y.**

Thus, the similarity measure s is given by the following formula:

$$\forall x,y \in N : s(x,y) = \frac{2a + b + c}{2(a + b + c)}$$

We can deduce the measure of dissimilarity from the following formula:

$$\forall x,y \in N : d(x,y) = 1 - \frac{2a + b + c}{2(a + b + c)} = \frac{b + c}{2(a + b + c)}$$

Besides this general form, there are also other forms of similarity measures such as binary data [14, 15, 17].

2.2 Profile concept

The operational adaptation is a measure which allows the implementation of actor oriented system in the specific developer context. This vision is in order to adapt the operation of a system in a specific context. In this context, it is possible to retain a practical dimension relative to a particular situation. We can globally set the profile development in the development activity context, as all the dimensions that allow to describe the static and dynamic aspects of the considered entities.

Modeling development profile aims to represent the static aspect of its basic entities and consider its dynamic aspect by its evolution over time. These two parameters determine its behavior during a work session, its guidance need and information as well as interaction support with the development system used [18, 19]. In this perspective, our goal is to provide a descriptive structure of the actor, commonly known profile, which proscribes the adaptation of the guidance system provided to the actor.

According to the needs, the profile data are shown differently. In general, we use a table of attribute-value pairs where each pair represents a property of the profile. The properties can be grouped by categories. The values are either numeric type, alphanumeric or probability distributions. These values are correlated with the adaptive services to be provided to the developer. The profile evolution is linked to its adaptation to the development context evolution regarding the guidance needs over time. Depending on the adaptation degree of the system, a profile can be described directly by the actor (reflective profile) selected pre-existing profile (expert profile) or deduced by the used system (dynamic profile) and adapted to the current context.

2.1. Profile organization

A profile can be designed and defined according to different orientations. Also, it is possible to proceed to its decomposition as needed. It is organized by a hierarchical structure where each

class is represented by a node. A class is defined through a decomposable and extensive description of data. In the first step, the Profile is described conceptually. Each instance leads to an operational profile wherein we find descriptions of these classes constructed from a data category relative to the associated semantics [18].

According to an adapted guidance, the construction of the hierarchy entities and categories constitutes a set of context characteristics. It is organized in a hierarchical structure of concepts (categories), where each category represents the knowledge of any interest area of an actor. Thus, the ratio of generalization/specification occurrence in this kind of structure provides a more realistic representation of the actor profile [18, 19]. Then, it is possible to consider only part of the profile by limiting only the current development context entities. This could be important in the perspective of a profile specialization.

The operational profile should keep the existing hierarchical organization; it can't be evaluated or interpreted only according to the specificity of an active development context. The specific context creation generates an instantiation of the profile while maintaining the hierarchical structure of the selected nodes [18, 19]. Each node represents the considered entity characteristics. Each characteristic is selected by evaluating its interest in the current context respecting the aim of the guidance provided to the actor.

2.2. Specific profile to software process

The profile concept is used in many areas in computer science. These later allow to develop the specific assistance services adapted to specific context. Among these profiles, we state: user account profile in the operating systems, the actor preferences profile in the learning systems and the developer and development team profile in a software process.

Based on these profile fundamental concepts and any variations that may characterize the different entities in a development context, we have developed within our framework, a generic profile model specifically dedicated to software process. This generic profile model serves as a basis for any specific profile definition to any particular development context. It is based on the following six dimensions: static and dynamic aspect of the main considered entities: activity, developer and development team. Our hierarchical generic profile is represented in the following figure.

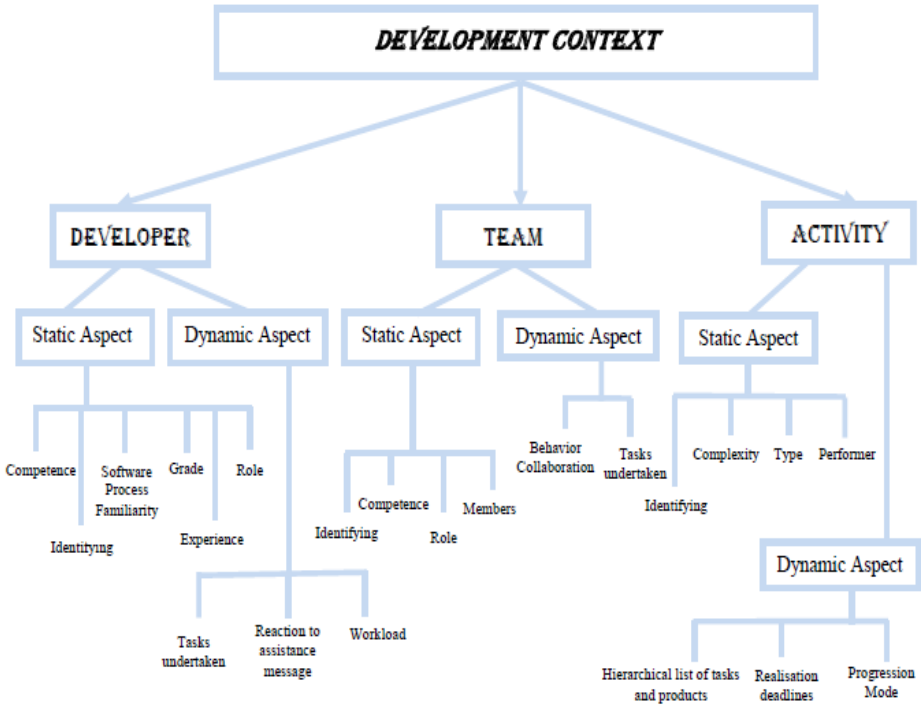

Figure 2. : Generic profile of development context.

3. SIMILARITIES IN THE SOFTWARE PROCESS

The proposed guidance system [3] addresses multiple views providing assistance to stakeholders. Our approach aims to optimize the profile classes. To be adaptive to both the context and identified needs, our model of adaptive guidance covers two levels of abstraction. It is based on a set of task and activity model, the model developer and development team, as well as the selection criteria specified by the mode of access for responding the objects of support by the defined assistance interventions (Figure 3.). The instantiation of this system is through rules of assistance detailed with the requirements for initiating appropriate actions to support a particular context [2, 3, 5].

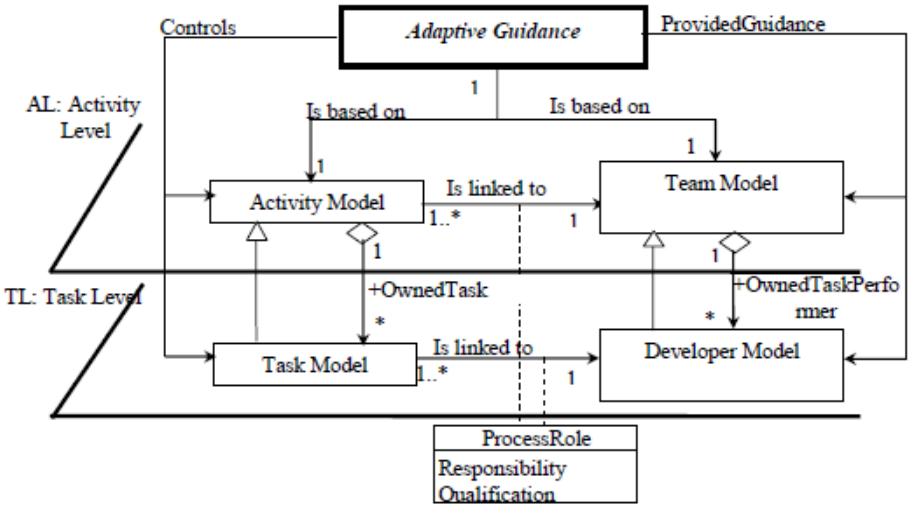

Figure 3. : Adaptive Guidance Model.

3.1. The Adaptive guidance model

This assistance system is based on the major models: the activity model, the developer model and the team development model.

a) The activity model: models the workflow, it is defined by:

- A hierarchical list of tasks,
- A mode of progression in the activity ensuring that all tasks can be performed under control in a preset order established by the designer,
- A temporal mode of progression specifying deadlines for completion.

The aspects of the activity model are useful for the assistance system to provide assistance on contextual growth in activity.

b) The developer model: defines the specific properties of each user. It allows our model to make adaptation according to these properties while maintaining the activity model. These properties can be either static or dynamic.

- *The static aspect refers to the user characteristics:*
  - his expertise in the field,
  - his familiarity with the software process model or with the software process,
  - his role in the activity.
- The dynamic aspect refers to the behavior of using the assistance system, the assistance system must be interpreted during the use of the process or system software support, for example :
  - the fact to execute, to define or to complete resource of software process,
  - the workload of an activity,
  - his reaction to a message of support.

c) ***The development team model:*** development environments allow exchanges and collaborative work. The assistance system can then construct a development team model that represents elements of the team. Example: trace of the various activities of the team as well as different interactions allow the developer to have a script about his own progress in the activity and the progression of the team. The properties of this model can be static or dynamic order.

- The static dimension references skills and team performance in the field of collaboration and distribution of task.
- The dynamic dimension deals with the behavior of the development team. It describes the actions taken by the team during the course of software process.

These data constitute indications that can be interpreted on the use of the assistance by the developer.

3.2. The assistance intervention

During the construction or interpretation of a software process model, the proposed model for assistance allows the developer to choose various support functions, namely:

3.2.1. Controlling and taking corrective initiative: protect the user of his own initiatives when they are inappropriate, inadequate initiative under progress.

3.2.2. Controlling and taking constructive initiative: the ability to take positive initiatives, executing and combining the performance of operations without user intervention.

3.2.3. Specific assistance: analyze the impact projection to avoid deadlocks or delays.

3.3. The profiles categorization

For the sake of productivity and optimal lead time, we were led to define an effective process for allocating appropriate guidance's. This efficiency is based on the process of maximizing the number of profiles classes to be considered in a development system. We will present our analysis of similarity and classification of our population.

The conception of the processing system will be done through various algorithms. They process both similarity index and hierarchical classification threshold for different profiles. To avoid an important dissemination of similarity, this classification will be ordered by level of similarity index. This classification will serve as the basis for the selection and assignment of appropriate types of assistance.

To reach an objective comparison between the profiles, an operator should be used for calculating similarity based on the instantaneous evaluation of selected features and associated weights. Despite the fact that this evaluation is not formal, it remains a crucial step for the classification. For this, we use the notion of symbolic learning for instant evaluation of some attributes of the profile as the behavior of the developer or development team.

3.3.1. Algorithm for computing Similarity Index

The evaluation of characteristics is based on the evolution of developer productivity. The weight value of each feature indicates the degree of its importance. The approach used for the evaluation of characteristics is based on "*COCOMO II"* work [20, 21]. It should be noted that our approach

considers a partially defined profile. The table of weights could be refined as soon as we have more data.

Consider two people profiles symbolized by:
$\mathbf{X} = (x_1, x_2, x_3, \ldots\ldots\ldots\ldots, x_n)$
$\mathbf{Y} = (y_1, y_2, y_3, \ldots\ldots\ldots\ldots.., y_n)$

Each characteristic is related to a weight representing its impact in the degree of similarity and symbolized by: $\mathbf{W_i} = (w_{i1}, w_{i2}, w_{i3}, \ldots\ldots\ldots\ldots.., w_{in})$

Example: a family of two profiles is semantically described in table 4 and table 5. The semantics evaluation and the weighting are determined by the project manager on an ongoing project [3, 20, 21].

Table 5. the profiles evaluation.

	Features	Evaluation of profile 1	Evaluation of profile 2
Model of activity	***Density of tasks in the activity***	High	Medium
	Complexity level	Medium	----------
	Activity Type	Tolerance zero	Margin Free
Model of developer	***Role***	Critical	Classic
	Competence	High	Medium
	Familiarity with Process Software	Medium	Low
	Behavior for assistance	Adequate	Acceptable
Model of developm ent team	***Skill Area Collaboration***	High	Medium
	Behavior for assistance	Acceptable	----------

To scan the semantics evaluation, we associate the weight corresponding to the consideration according to each attribute.

Table 6. Table of weights

W [1]	P2
W [2]	P2
W [3]	P3
W [4]	P2
W [5]	P2
W [6]	P2
W [7]	P2
W [8]	P2
W [9]	P4

With $[i] \in [1, 5]$. Where Pi represents the computing value

The algorithm processing is as follows: For any feature, whether it is identical to the profiles, we increment the similarity index by the weight of this feature, otherwise, if the difference between the two characteristics is < 1/2, we add half the weight of it, otherwise, we move to the next feature.

The value 1/2 represents the average distance between two successive levels of an attribute evaluation.

After all iterations, the similarity function obtained is formalized as follows:

$$S(x, y) = \frac{A(x, y)}{\sum W[i]}$$

With:
- X and Y represent the characteristics of the two profiles.
- W [ ] represents the weighting of each feature.
- A (x, y) represents the sum of the weights between the two profiles, it is included between 0 and ΣW[i].

The similarity function developed verifies the properties of a similarity measure.

∀x, y two profiles ∈ N

- ***If the profiles (x, y) are identical***
  Then A(x, y) = A(x, x) = ∑ W[i] with i= 1..9 ⇒ S(x, y) = A(x, y) / ∑ W[i] = 1
- ***If the profiles (x, y) are totally different (the values are all above features > 1/2) or feature is not defined in x and/or in y. Then for any characteristic A (x, y) = 0 ⇒ S(x, y) = A(x, y) / ∑ W[i] = 0.***
- ***If any profiles are neither identical nor different then " It is appropriate to consider three subsets possible through the following 03 cases":***

(1) $A(x, y) = 0$ *for features with a difference* $> 1/2$ *or feature is not defined in x and/or in y.*

(2) $A(x, y) = \Sigma\, 1/2 W[i]$ *for the characteristics with a difference* $< 1/2$.

(3) $A(x, y) = \Sigma\, W[i]$ *for completely identical characteristics.*

Finally for all characteristics:

$$A(x, y) = A(x, y)_{(1)} + A(x, y)_{(2)} + A(x, y)_{(3)}$$

$$= 0_{(1)} + 1/2 \sum W[i]_{(2)} + \sum W[i]_{(3)}$$

Then

$$S(x, y) = A(x, y) / \sum W[i] = (0_{(1)} + 1/2 \sum W[i]_{(2)} + \sum W[i]_{(3)}) / (\sum W[i]_{(1)} + \sum W[i]_{(2)} + \sum W[i]_{(3)}) \leq 1.$$

This allowed us to affirm that:
- ✓ ∀ x, y two profiles ∈ N, The similarity function S(x, y) ∈ [0, 1].
- ✓ ∀ x, y two profiles ∈ N, then S(x, x) = S(y, y) ≥ S(x, y).

```
Input: The Profiles of Two Developers X and Y.  - Table Weights W [].
Output: The value of similarity index S (x, y) ∈ [0, 1].

Begin:
   A(x, y) = 0; // Similarity index between X and Y
   For all characteristics Xi and Yi Do
      If both characteristics are identical
         Then A(x, y) = A(x, y) + W[i];
         Else
            If (| (Xi – Yi) | < 1/2)
               Then A(x, y) = A(x, y) + 1/2W[i];
            End if;
      End if;
End.
```

Figure 4. : Algorithm for calculating similarity index.

Example: based on the assessing approach of the COCOMO model, the quantification of each characteristic of a profile P is on the data range] 0, 2 [. It is usually done through three steps, described by high, medium or low levels contribution, applying the following rules:

1: impact of middle order.
<1: positive impact.
>1: negative impact.

Table 7. The profiles evaluation

	Features	profile 1	profile 2	profile 3	profile 4
Model of activity	*Density of tasks in the activity*	1.65	1.20	1.10	1.65
	Complexity level	1.00	0.60	1.00	1.00
	Activity Type	1.70	1.20	1.20	1.70
Model of developer	*Role*	1.15	0.70	1.15	1.60
	Competence	055	1.00	1.00	0.55
	Familiarity with Process Software	0.40	----	0.35	0.40
	Behavior for assistance	0.40	0.60	0.40	0.40
Model of developme	*Skill Area Collaboration*	0.70	0.70	0.70	0.70
	Behavior for assistance	0.95	----	0.95	0.95

The weight value of each feature indicates the degree of its importance. The project manager associates the value correspondence table of weights, for example.

Table 8. Table of weights

W [1]	1
W [2]	1
W [3]	2
W [4]	1
W [5]	1
W [6]	1
W [7]	1
W [8]	1
W [9]	2

Based on our approach, the calculation of the similarity value between profiles is given by:

Table 9. The similarity values

	Similarity value
S (P_1, P_2)	**0.31**
S (P_1, P_3)	**063**
S (P_1, P_4)	**0.95**
S (P_2, P_3)	**0.55**
S (P_2, P_4)	**0.27**
S (P_3, P_4)	**0.59**

3.3.2. Ascending Hierarchical Classification Algorithm for (addressing) and similarity threshold

Classifying is grouping objects together according to similar criteria. There are two main families of classification techniques:

- The non-hierarchical classification or partitioning leads to the decomposition of the set of all individuals in m disjoint sets or equivalence classes, the number of classes is fixed for m.
- The hierarchical classification for a given accuracy, two individuals may be confused in the same group, whereas in a higher level of accuracy, they will be separated and belong to two different subgroups.

We opted for the hierarchical classification in increments of similarity that led to construct a classification tree showing the transition profiles to the group through a series of consolidation.

The obtained classification is related to the variables selected to describe individuals, in our case the developers. They are called the active variables, which will be based on the classification of individuals. For this, and to avoid dispersion of profiles similarity, the user must set the level of similarity describing each time the similarity values to consider and the level of precision represents the similarity threshold to be applied on the profiles of guidance to classify.

Input: Threshold Similarity Kmin; Profiles guidance X1, ……, Xn.
Output: Different classes C1, ……, Cm.

Begin

1- Create the similarity matrix (NXN),

Linking the N guidance profiles considered by the indices of relative similarities.
On each level: K = min field similarity to consider

2- Search the largest index of similarity in Mij.
3- If (S(Xi, Xj) ≥ K) Then Put (Xi, Xj) in the same class Cm.
4- Remove the rows and columns of Xi and Xj the matrix Mij.
5- Add Cm to the matrix M given that:
S(Xi, Cm) = S(Cm, Xi) = Min ((S(Xi,Xj));
Go to step 2;
Else if kmin < K (Min current level) Then set the new level
Else Stop.

End.

Figure 5. : Ascending Hierarchical Classification Algorithm

3.3.3. Processing of the algorithm on a sample guidance profiles

We have the similarity threshold set and profiles guidance X1, X2, **…….,** Xn as input. Our algorithm will create a square matrix of size (NXN) considering the number of profiles to classify and index of similarity between profiles. See the following graph of similarity:

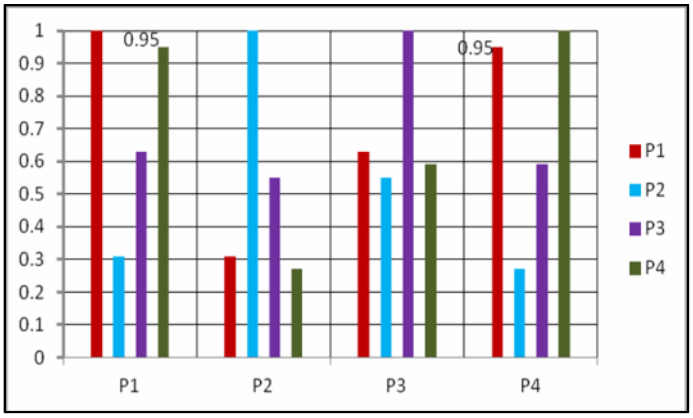

Figure 6. : The graph of similarity

For a level of 0.2 and a minimum similarity threshold of 0.50, set initially by the user, which fixes K to 0.80, the application of this algorithm is illustrated as follows:

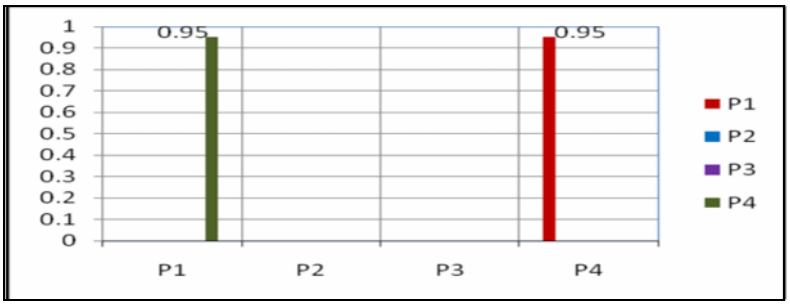

Figure 7. : Illustration of the algorithm

The maximum value of similarity in this table is 0.95, it is the index of similarity between two profiles P1 and P4, and these profiles will be aggregated to the first group C1.

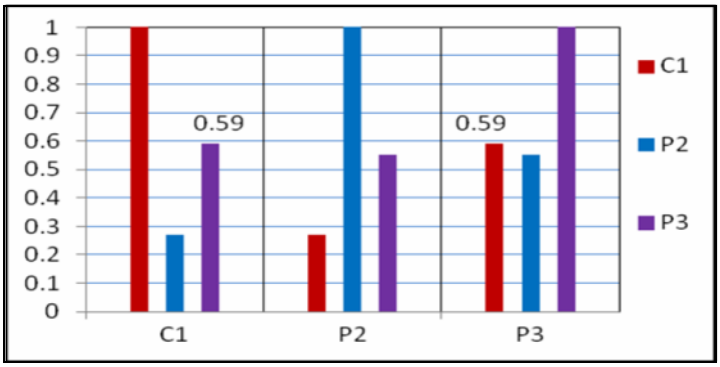

Figure 8. : Classification (first set)

At this level, we find that the similarity indices are all below the minimum level of similarity of the first set.

- Since Kmin is less than K, we fix the next level, for this example, the new K will be set at 0.50. It repeats the previous steps until all the indices similarities are below the new threshold of similarity.
- The maximum value of similarity in the new similarity matrix is 0.59; it is the index of similarity between two profiles C1 and P3, and the two profiles will be aggregated to the first group C1 and the profile P2 constitute the class C2.

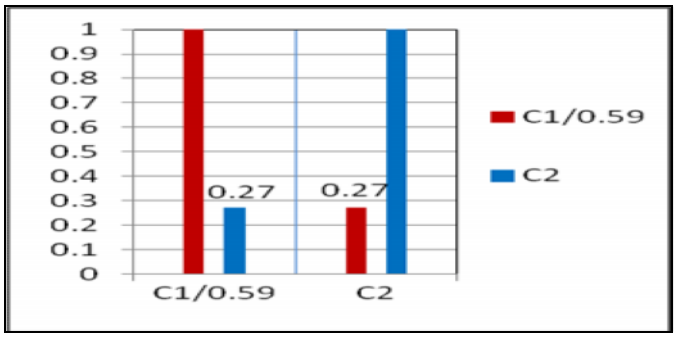

Figure 9. : The major classification

The application of our approach to similarity index and hierarchical clustering allowed us to deduce of the initial profiles number, two major classes C1, C2. This will provide the basis for optimal allocation of the appropriate guidance.

The guidance profile (GP) associated to each profile (Px) class is based on the following formula:

$$GP\ (Px) = \sum A_i W_i / 2 * \sum W_i \quad avec\ i=1\ to\ 9$$

With:

A_i : the defined characteristic value.
W_i : the associated weight.
In our case, the guidance profile of each class based on the smallest similarity value is given by:

Table 10. The associate guidance value

	C1	C2
GP	***0.53***	***0.33***

It should be noted that the value of GP is ranged from 0 to 1. The range associated with each type of guidance is defined by the project manager. For instance, if the range of the corrective guidance is bounded between 0 and 0.40 and the range of the constructive guidance is between 0.41 and 0.70, we automatically associate a corrective guidance to the class C2 and a constructive guidance to the class C1.

4. CONCLUSION

The system presented in this paper is an approach based on similarity to process guidance model in the software process. It allows the profiles optimization, ie: classes presented through the semantics description of handled data, the definition of a system for processing the similarity index and classification of guidance profiles. The aim of this work is to facilitate the analysis of our population using the adaptive development context involved in the execution of a software process.

The system has been designed and implemented and its practical assessment seems to be promising with a significant impact on the productivity of software process development.

In perspective and in order to improve this approach, it would be interesting to develop a similarity measure that takes into account the partial knowledge of profile characteristics. This allows the selection of a profile as the "best effort".

REFERENCES

[1] Kirk, D.C, MacDonell, S.G., & Tempero, E. 2009 Modeling software processes - a focus on objectives, in Proceedings of the Onward, 2009. USA, ACM Press, pp.941-948.

[2] Ivan Garcia and Carla Pacheco. 2009. Toward Automated Support for Software Process Improvement Initiatives in Small and Medium Size Enterprises. Book chapter. Software Engineering Research, Management and Applications 2009. Volume 253, pp. 51–58. Springer-Verlag Berlin Heidelberg. ISBN: 978-3-642-05440-2.

[3] Hamid Khemissa, Mohamed Ahmed-Nacer, Mourad Daoudi, 2008. A Generic assistance system of software process. In proceeding of the IASTED International Conference on Software Engineering:

Software Engineering. (SE '08), ACTA Press, Anaheim, CA, USA, 237-242 ©2008 February., Austria.

[4] Dadam, P. and Reichert, M. 2009. The ADEPT Project: A Decade of Research and Development for Robust and Flexible Process Support – Challenges and Achievements,' Springer, Computer Science - Research and Development, 2009. Vol. 23, No. 2, pp. 81-97.

[5] MALGOUYRES H., MOTET G. 2006. A UML Consistency Verification Approach Based on Meta modeling Formalization. Symposium on Applied Computing, Dijon, France, ACM publishers.

[6] C. EyssautierBavay, 2008. Modèles, langage et outils pour la réutilisation de profils d'apprenants", Thèse de doctorat de l'Université Joseph Fourier Grenoble 1, 26 Mai 2008.

[7] CHU09 CHUNG-FOO-WO Daniel. 2009. Adaptation dynamique par tissage d'aspects de l'optimisation. Thèse de doctorat, Université de Nice - Sophia Antipolis, 5 Mars 2009, Equipe RAINBOW, Pôle GLC.

[8] S. dami et al, 1998. APEL: a Graphical Yet Executable Formalism for Process Modeling. Kuwler Academic Publisher, pp. 60-96, Boston, January 1998.

[9] Borislava I. Simidchieva, Lori A. Clarke , Leon J. Osterweil. 2007. Representing process variation with a process family. ICSP'07 Proceedings of the 2007 international conference on Software process. Springer-Verlag Berlin, Heidelberg © 2007 ISBN: 978-3-540-72425-4.

[10] Coulette Bernard., Crégut Xavier. et al, 2000. RHODES, a Process-centered Software Engineering Environment, in Proc. of ICEIS 2000, Stafford, pp 253-260, 2000.

[11] Bradshaw J., Editor,. 2000. Handbook of Agent Technology. MIT Press, 2000.

[12] OMG. Inc. 2008. Software Engineering Meta-Model Specification version 2.0: Formal/2008-04- 01.

[13] Vivien Robinet, Gilles Bisson, Mirta B. Gordon, Benoît Lemaire. 2007. Induction of High-level Behaviors from Problem-solving Traces using Machine Learning Tools. Published in "IEEE Intelligent Systems 22, 4 (2007) 22".

[14] Ahmed Belkhirat, Abdelghani Bouras, Abdelkader Belkhir. 2009. A New Similarity Measure for the Anomaly Intrusion Detection. In Third International Conference on Network and System Security (NSS).

[15] Ahmed Belkhirat, Abdelkader Belkhir, Abdelghani Bouras 2011 A New Similarity Measure for the Profiles Management, UKSIM '11: Proceedings of the Tenth International Conference on Computer Modeling and Simulation, Cambridge England.

[16] Xavier Aimé, Frédéric Furst, Pascale Kuntz, Francky Trichet. 2009. «SEMIOSEM: A Semiotic-Based Similarity Measure". On the Move to Meaningful Internet Systems (OTM 2009), International Workshop on Ontology content and evaluation in Enterprise (OntoContent'2009), Lecture Notes in Computer Science-LNCS 5872, pp. 584-593. Springer-Verlag (Berlin Heidelberg). ISBN 978-3-642-05289-7. Villamoura, Portugal.

[17] S. Boriah, V. Chandola, and V. Kumar. 2008. Similarity measures for categorical data: A comparative evaluation. In SDM 2008: Proceedings of the eighth SIAM International Conference on Data Mining, pages 243-254, 2008.

[18] Zemirli W.Nesrine. Vers le développement d'un système de recherche d'information personnalisé intégrant le profil utilisateur. Université Paul Sabatier & Institut National Polytechnique de Toulouse (IRIT 2004). Formation Doctorale en informatique Année 2004.

[19] S. Gauch, M. Speretta, A. Chandramouli, and A. Micarelli. User profiles for personalized information access. The Adaptive Web, 4321 :54.89, 2007.

[20] Barry W. Boehm, Chris Abts, A. Winsor Brown, Sunita Chulani, Bradford K. Clark, Ellis Horowitz, Ray Madachy, Donald J. Reifer, Bert Steece. 2009 Software Cost Estimation With COCOMO II. Prentice Hall Edition, ISBN: 0137025769, 978013702576.

[21] Kirk, D., & MacDonell, S. 2009. A simulation framework to support software project (re)planning, in Proceedings of the 35th Euromicro Software Engineering and Advanced Applications (SEAA) Conference. Patras, Greece, IEEE Computer Society Press, pp.285-292.

Authors

Mohamed Ahmes-Nacer is a full Professor at USTHB (Algiers's University). He is the Director of Computer Engineering Laboratory at USTHB and is in charge of the software engineering team. He published extensively and his current research interests include process modeling, information systems, software architecture based components and service web development.

Hamid Khemissa is a full associate professor at Computer Systems Department, Faculty of Electronics and Computer Science, USTHB University, Algiers. He is member of the software engineering team at computer system laboratory LSI, USTHB. His current research interests include Software Process Modeling and Software Modeling Assistance.

Pr. Abdelkader Belkhir (PHD Paris 6) is a full Professor at the computer science department, faculty of electronics and computer science, USTHB University, Algiers. Currently, he leads the team multimedia and computer security at LSI laboratory, USTHB. His research field includes multimedia, web services and computer security.