*JKAU: Comp. Sci.*, **Vol. 5**, pp: 75 - 81 (2016 A.D./ 1437 A.H.)
DOI: 10.4197 / Comp. 5 - 5

# A Cognitive Attraction Network Approach to the Software Team Building Decision Problem

**Omar Elshinnawey, Ghada Abuguyan,** and **Zohair Chentouf**

*Software Engineering Department, College of Information and Computer Sciences, King Saud University, Riyadh, KSA*

*zchentouf@ksu.edu.sa*

*Abstract.* Software team selection can be done based on several criteria. One among them is the individual developers' productivity. However, the sole productivity cannot be accurate if one has to take into consideration the goals of the software project. Alignment of the productivity-based decision with the project's goals is the aim of the present article. The proposed automatable solution is based on Cognitive Attraction Networks.
*Keywords*: Team Building; Software Engineering; Project Management; Cognitive Attraction Networks.

## 1. Introduction

The optimization of teamwork has been intensively studied in various areas[1-4]. The early interest in software teamwork first aimed at the structure of software teams[5-6] then researchers began to investigate on the software team performance factors, like in[7-9]. In[9], for instance, emphasis is put on the developers' personality, the work environment, the structure of the team, leadership, and communication. The impact of the team performance on the software project's goal achievement also has been studied. For example,[10-12] highlighted that impact in terms of productivity, product quality, and project success. Study of the emergence of individual developers' factors at the team's performance level was performed in[13-16].

The present investigation's first objective is an automatic aid to software development team building based on individual developers' performance. A second objective is to align the team member selection with the goals of the software project. The direct correlation between developer assignment and the software production success was highlighted in [17, 18]. In general, it is up to the project's leaders to perform the assignment based on their experience of people, software constraints (*e.g.* reliability), and the project skill requirements. This task becomes cumbersome with medium and large size projects. That is because the number of possible combinations of the candidate software developers' roles rapidly degenerates into a too large solution space. This makes it virtually impossible to put the hand on an optimal solution. Hence the need for a decision support system to help tackle the problem's complexity. Besides the complexity of the problem, another research motivation comes from the fact that well reputed software process models like People - CMM[19], Personal Software Process – PSP[20], Team Software Process – TSP[8], and the Rational Unified Process – RUP[21], do not model the sub-process of developer assignment. Our research is also motivated by a third fact; there are only a few

related works to ours. Ngo-The and Ruhe[22] proposed a method for assigning developers to software projects by decomposing a project to a series of releases. The assignment problem is solved for every release based on the developers' competencies. However, the method is only applicable to mature software teams where the processes are fully defined, measured, and controlled. Tsai *et al*.,[23] proposed a method for selecting software developers which focuses on human and material resources rather than task scheduling. The method is meant to be used under dynamic and stochastic constraints. It considers two factors: the developer's programming productivity and salary. Otero, *et al*.,[24] proposed a process to assign tasks to software developers while they do not completely fulfill the project required skills. The process takes into consideration the actual skills, the required expertise levels, and the priority of the tasks to perform.

At the best of our knowledge, there is no previous research work that addressed the problem of software task assignment under the constraint of fulfilling the project's goals. This is the aim of our research. First, developers' competences are mined from the bug-tracking system content related to previous projects. We then build a Cognitive Attraction Networks-based model which encompasses the developers' competences and the project's skill requirements and goals. The assignment algorithm is described along with a case study.

Section 2 of the present article suggests a method for mining developers' competency from a bug-tracking system. Section 3 presents in detail the proposed task assignment algorithm. A case study is presented in Section 4. Section 5 concludes the work.

## 2. Mining Developers' Competency

Developers' competency needs to be known before assigning the new project's tasks to them. We assume that all the candidate developers were involved in past software projects. The bug-tracking system should contain information about the programming and debugging tasks that belong to the past projects. We mine this pool of information to conclude the competency level of every developer. For this aim, we assume the following features that are present in popular bug-tracking systems, like[25-26]:

- A bug-tracking system contains defect and task descriptions.
- Every defect or task was assigned to a single developer.
- Every defect or task had a start date, a desired end date, and an actual end date.

We consider the software developers' competences listed in[27-29]:

- Reasoning: the process of forming conclusions or inferences from facts or premises.
- Decision making: the ability to make the right decision on the right time.
- Judgment: to identify the right attributes of an existing software or human mechanism.
- Stress tolerance: the ability to cope up with work pressure.
- Openness to change: self-improvement with present and future software business needs.
- Teamwork and cooperation: the ability to interact with colleagues and cooperate with them.

Other competences need to be added as well:

- Problem solving: the ability to solve unexpected technical problems and manage possible risks.
- Required technical hands-on skills like Java, Linux, SQL, setting a small network for tests, etc.
- Duration estimation: the ability to deliver assigned tasks in time.

We assume that the project manager or any allowed team member has the responsibility of calculating
the candidate developers' competences. For every candidate $D_j$ the manager:

a) Selects a task $T_i$ that was performed by $D_j$ for some past projects.
b) Calculates the actual duration of the task as:

$$Actual\ Duration = End\ Date - Start\ Date$$

c) Determines the list of required competences and rates the difficulty degree of each on a scale of 1-5. For the reasoning competence, for example, this will be denoted *required reasoning score*.
d) Assesses the performance of the developer $D_j$ who was in charge of $T_i$ in

regard to every competence. For the reasoning skill, for instance, this will be denoted *actual reasoning score*, which is an integer number on the scale 1-5.

e) Calculates the performance factors of the developer $D_j$ in regard to the task $T_i$. We adopt the following metrics, which is based on the above listed competences:

- Duration Deviation (DD):

$$DD = \frac{actual\ duration\ -\ duration\ estimate}{duration\ estimate}$$

- Reasoning Deviation (RD):

$$RD = \frac{actual\ reasoning\ score\ -\ required\ reasoning\ score}{required\ reasoning\ score}$$

- Decision Making Deviation (MD):

$$MD = \frac{actual\ decision\ score\ -\ required\ decision\ score}{required\ decision\ score}$$

- Judgment Deviation (JD):

$$JD = \frac{actual\ judgment\ score\ -\ required\ judgment\ score}{required\ judgment\ score}$$

- Stress tolerance Deviation (SD):

$$SD = \frac{actual\ stress\ tolerance\ score\ -\ required\ stress\ tolerance\ score}{required\ stress\ tolerance\ score}$$

- Openness to change Deviation (OD):

$$OD = \frac{actual\ openness\ to\ change\ score\ -\ required\ openness\ to\ change\ score}{required\ openness\ to\ change\ score}$$

- Team Work and cooperation Deviation (TWD):

$$TWD = \frac{actual\ team\ work\ score\ -\ required\ team\ work\ score}{required\ team\ work\ score}$$

- Problem solving Deviation (PD):

$$PD = \frac{actual\ problem\ solving\ score\ -\ required\ problem\ solving\ score}{required\ problem\ solving\ score}$$

- Technical Skill X Deviation ($X_D$):

$$X_D = DD\ \frac{5}{diff}$$

where: X ∈{Java, PHP, ...}; 1≤ diff ≤ 5 is the difficulty rate of the required skill X.

f) Repeats steps a)-e) for every task that was performed by the developer $D_j$.

g) Calculates the competence gap of $D_j$ with regard to every one of the above listed skills as the mean of that skill's deviations. For example, the reasoning gap:

$$RG = Mean\_RD$$

where *Mean_RD* is the mean of RD of all the tasks performed by the developer $D_j$. Similarly, all the other competence gaps will be calculated, namely: Duration Gap (DG), Decision Making Gap (MG), Judgment Gap (JG), Stress tolerance Gap (SG), Openness to change Gap (OG), Team Work and cooperation Gap (TWG), Problem solving Gap (PG), and Technical Skill X Gap ($X_G$).

## 3. Software Task Assignment Decision Problem and Solution

Recall the problem being tackled by the present research. Given a new software development project, the related list of software development tasks, the list of competences required for every task, the list of candidate developers along with their competence levels mined from past projects' data (see Section 2), and the list of goals to fulfill by the project, the problem to solve is one of assigning the best candidate developer to every development task in a way that maximizes the goal satisfaction and takes into consideration the competence requirements of the task and the corresponding actual competence levels of the developers. Note that a goal can be a non-functional requirement, such as reliability, efficiency, usability, availability, portability, maintainability, *etc*., or a project constraint such as time or budget. Formally, a software task assignment decision problem constituents are: a set of tasks $T_i$, i=1, …, NT, a set of goals $G_k$, k=1, …, NG, and a set of developers $D_j$, j=1, …, ND. Every goal $G_k$ has an importance weight $p_k$∈[0, 1], with $\sum_{k=1}^{NG} p_k = 1$. Every task $T_i$ contributes with a weight $w_{ki}$∈[0, 1] in satisfying the goal $G_k$, with $\sum_{i=1}^{NT} w_{ki} = 1$ . Every task $T_i$ requires a competence a set of competences $C_u$, u=1, …, NC (see Section 2) with an importance level $r_{iu}$∈[0, 1], $\sum_{u=1}^{NC} r_{iu} = 1$. The competence gap of a developer $D_j$ with regard to a competence $C_u$ will be denoted $cg_{uj}$∈[-0.8, 4]. The latter interval values are engendered by the fact that skill deviations in Section 2 were based on a 1-5 scale.

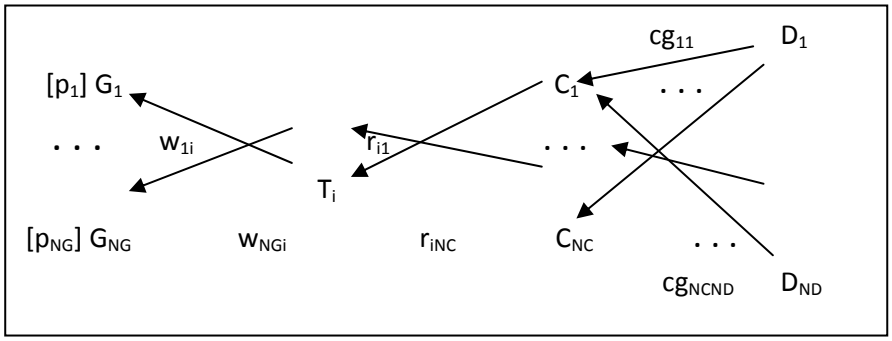


**Fig.1. CAN of the Software Task Assignment Decision Problem.**

We use these formulation elements to construct a Cognitive Attraction Network (CAN). CAN was introduced in [30-31]. Figure 1 depicts the developer assignment problem's CAN with a single task. The assignment problem can be formulated as selecting a developer $D_j$ among the set of ND candidate developers to be in charge of a given task $T_i$ under the constraint of fulfilling the goals $G_1$, …, $G_{NG}$. For this aim, we need to calculate the so called cognitive attraction [30] of the goals made on developers through the following steps:

a) The attraction of the task $T_i$ performed on every developer $D_j$:

$$t_{ij} = \sum_{u=1}^{NC} r_{iu} * cg_{uj}$$

b) The attraction of every goal $G_k$ performed on every developer $D_j$:

$$g_{kj} = p_k * w_{ki} * t_{ij}$$

c) The sum of goal attractions made on the developer $D_j$:

$$g_j = \sum_{k=1}^{NG} g_{kj}$$

Since $g_j$ is a function of the developer's skill gaps $cg_{uj}$, the developer with the smallest value of $g_j$ is selected for the task $T_i$.

d) Repeat steps a)-c) for every other task $T_v$ excluding all the developers $D_t$ who were selected for previous tasks.

## 4. Case Study

Let us take an illustrative example. Suppose that we have three developers: $D_1$, $D_2$, and $D_3$, one task $T_i$, three required competences $C_1$, $C_2$, and $C_3$, and four goals $G_1$, $G_2$, $G_3$ and $G_4$. The assignment algorithm inputs and processing results are summarized in Table 1-Table 6.

**Table 1. Importance Weights of Goals ($p_k$)**

| $G_1$ | $G_2$ | $G_3$ | $G_4$ |
|---|---|---|---|
| 0.150 | 0.150 | 0.300 | 0.400 |

**Table 2. Contributions of The Task $T_i$ IN Goals ($w_{ki}$)**

| | $G_1$ | $G_2$ | $G_3$ | $G_4$ |
|---|---|---|---|---|
| $T_i$ | 0.300 | 0.400 | 0.200 | 0.500 |

**Table 3. COMPETENCES' IMPORTANCE WEIGHTS FOR THE TASK $T_i$ ($r_{iu}$)**

| | $C_1$ | $C_2$ | $C_3$ |
|---|---|---|---|
| $T_i$ | 0.200 | 0.300 | 0.500 |

**Table 4. DEVELOPERS' COMPETENCE GAPS ($cg_{uj}$)**

| | $D_1$ | $D_2$ | $D_3$ |
|---|---|---|---|
| $C_1$ | -0.200 | 1.500 | -0.750 |
| $C_2$ | 0.250 | -0.500 | 0.670 |
| $C_3$ | 0.670 | 0.250 | 0.670 |

**Table 5. ATTRACTION OF $T_i$ ON DEVELOPERS ($t_{ij}$)**

| | $D_1$ | $D_2$ | $D_3$ |
|---|---|---|---|
| $T_i$ | 0.370 | 0.275 | 0.386 |

**Table 6. ATTRACTION OF GOALS ON DEVELOPERS ($g_{kj}$ and $g_j$)**

| | $D_1$ | $D_2$ | $D_3$ |
|---|---|---|---|
| $G_1$ | 0.017 | 0.012 | 0.017 |
| $G_2$ | 0.022 | 0.017 | 0.023 |
| $G_3$ | 0.022 | 0.017 | 0.023 |
| $G_4$ | 0.074 | 0.055 | 0.077 |
| Sum | 0.135 | 0.100 | 0.141 |

Based on the results of Table 6, the best candidate for the task $T_i$ is the developer $D_2$.

## 5. Conclusion

This paper presented a Cognitive Attraction Network-based algorithm destined to support the project manager in team selection under the constraints of software project's goals, skill requirements, and candidate developers' actual competences. These competences need to be mined from the bug-tracking system of the company. Specifically, the data related to the past tasks completed by the candidate developers are to be exploited to infer their individual competence gaps. The proposed mining method can easily be automated as a feature of the bug-tracking system itself. In this way, the task of calculating the developers' competence gaps would be dramatically simplified. The proposed task assignment algorithm builds on the results of this competency calculation. The algorithm's output is a ranking of developers' suitability for every project's task.

As future work, there should be an empirical validation through real utilization and evaluation of the proposed solution.

### References


[1] **Cohen, S. G.** and **Bailey**, **D. E.,** "What Makes Team Work: Group Effectiveness Research from the Shop Floor to the Executive Suite," *Journal of Management*, **23**: 239-290, 1997.

[2] **Guzzo, R. A.** and **Dickson**, **M. W.,** "Teams In Organizations: Recent Research on Performance and Effectiveness," *Annual review of psychology*, **47**: 307-38, 1996.

[3] **Mathieu, J., Maynard, M. T., Rapp, T.** and **Gilson, L.,** "Team Effectiveness 1997-2007: A Review of Recent Advancements and a Glimpse into the Future," *Journal of Management*, **34,** (3): 410-476, 2008.

[4] **Rasch, R. H.** and **Tosi**, **H. L.,** "Factors affecting software developers' performance: An integrated approach," *MIS Quarterly*, **16**, (3): 395 - 413, Sept, 1992.

[5] **Baker**, **F.,** "Chief programmer team management of production programming," *IBM Systems Journal*, **11**, (1): 56-73, Jan. 1972.

[6] **Brooks**, **F.,** *The Mythical Man-Month*, Addison-Wesley, 1975.

[7] **Constantine**, **L.,** *Peopleware Papers: The Notes on the Human Side of Software*, Prentice-Hall, 2001.

[8] **DeMarco, T.** and **Lister**, **T.,** *Peopleware: Productive Projects and Teams*, 2nd ed. New York: Dorset House Publishing Co, 1999.

[9] **Shneiderman, B.,** *Software Psychology: Human Factors in Computer and Information Systems*, Winthrop Publishers, 1980.

[10] **Boehm**, **B. W.** *Software Engineering Economics*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1981.

[11] **Curtis, B., Hefley, W. E.** and **Miller**, **S. A.,** *People Capability Model (P-CMM),*Software Eng. Inst., Carnegie Mellon Univ., 2001, version 2.0, tech. rep. CMU/SEI-2001-MM-001.

[12] **McConnell, S.** "Problem programmers," *IEEE Software*, **15**, (2): 126–128, 1998.

[13] **Acuña, S. T. Gómez, M.** and **Juristo**, **N.,** "How do personality, teamprocesses and task characteristics relate to job satisfaction and software quality?" *Information and Software Technology*, **51**, (3): 627-639, March, 2009.

[14] **Bradley**, **J. H.** and **Hebert**, **F. J.,** "The effect of personality type on team performance," *Journal of Management Development*, **16**, (5): 337–353, 1997.

[15] **Guinan, P. J., Cooprider, J. G.** and **Faraj, S.** "Enabling software development team performance during requirements definition: A behavioral versus technical approach," *Information System Research*, **9**, (2): 101–125, June 1998.

[16**]** **Rajendran**, **M.,** "Analysis of team effectiveness in software development teams working on hardware and software environments using Belbin Self-perception Inventory", *Journal of Management Development*, **24**, (8): 738-753, 2005.

[17] **Campion, M. A. Medsker, G. J.** and **Higgs**, **A. C.,** "Relations between Work Group Characteristics and Effectiveness: Implications for Designing Effective Work Groups", *Personnel Psychology*, **46**, (4): 823 – 847, 1993.

[18] **Cohen**, **S. G.,** "Designing Effective Self-Managing Work Teams", In *M. Beyerlein ed., Advances in interdisciplinary studies of work teams*, vol. 1, Series of self-managed work teams, Greeenwich, Connecticut, JAI Press, 1993.

[19] **Curtis, B., Hefley, W. E.** and **Miller, S. A.,** *People Capability Model (P-CMM),*Software Eng. Inst., Carnegie Mellon Univ., 2001, version 2.0, tech. rep. CMU/SEI-2001-MM-001.

[20] **Silva, F. Q. B. da.** and **França, A. C. C.,** "Towards Understanding the Underlying Structure of Motivational Factors for Software Engineers to Guide the Definition of Motivational Programs", *Journal of Systems and Software*. Article in Press.

[21] *Diagnostic and Statistical Manual of Mental Disorders (DSM-IVTR*), Fourth ed., Text Revision, American Psychiatric Association, 2000.

[22] **Ngo-The, A.** and **Ruhe**, **G.,** "A systematic approach for solving the wicked problem of software release planning, *Soft Comput*ing, **12**, (1): 95–108, 2008.

[23] **Tsai, H., Moskowitz, H.** and **Lee**, **H.,** "Human resource selection for software development projects using Taguchi's parameter design",*European Journal of Operational Research*, **153**, (1): 167–180, 2003.

[24] **Otero, L.D., Centeno, G., Ruiz-Torres, A.J.,** and **Otero**, **C.E.,** "A systematic approach for resource allocation in software projects", *Computers & Industrial Engineering*, **56**, (4): 1333–1339, 2009.

[25] **BugZilla,** "BugZilla Features" [online] retrieved Januaery 2016 from http://www.bugzilla.org/features/.

[26] **Mantis**, "Mantis Feature List" [online] retrieved January 2016 from, http://www.mantisbt.org/wiki/doku.php/s:features.

[27] **Belkadi, F.** and **Bonjour**, **E.,** "Competency characterization by means of work situation modeling, Computers in Industry", **58**, (2): 164-178, 2007.

[28] **Draganidis, F.** and **Mentzas**, **G.,** "Competency based management: a review of systems and approaches", *Information Management & Computer Security*, **14**, (1): 51-64, 2006.

[29] **Acuña, S. T., Juristo, N.** and **Moreno**, **A. M.,** "Emphasizing Human Capabilities in Software Development", *IEEE Software*, **23**, (2): 94-101, 2006.

[30] **Chentouf, Z.,** "Cognitive Attraction Theory and Moral Judgment," *Psychology*, **4**, (1): 38-43, 2013.

[31**]** **Chentouf**, **Z.,** "Homo Informaticus", *L'Harmattan*, Paris, 2000.

# مقاربة لتشكيل فرق مبرمجين باستخدام شبكات الجذب المعرفي

**عمر الشناوي، وغادة أبوقيان، وزهير شنتوف**

*قسم هندسة البرمجيات، كلية علوم الحاسب والمعلومات، جامعة الملك سعود*
*الرياض، المملكة العربية السعودية*

*المستخلص.* تعتمد عملية اختيار أعضاء فريق إنجاز مشروع برمجيات على كثير من المعايير, من بينها إنتاجية المبرمجين. لكن الاعتماد على الإنتاجية وحدها يجانب الدقة في كثير من الأحيان, خصوصًا عند النظر إلى علاقتها بأهداف المشروع كافة. أول هدفين لهذه الورقة البحثية هو إيجاد نموذج مفهومي يربط بين القرارات المبنية على الإنتاجية وأهداف المشروع على اختلافها. وثاني الهدفين هو أتمتة النموذج باستعمال شبكات الجذب المعرفي.