Modeling Software Development Projects Using Fuzzy Cognitive Maps

Wojciech Stach
Department of Electrical and Computer Engineering
University of Alberta,
Edmonton T6G 2V4, Canada
wstach@ece.ualberta.ca

Lukasz Kurgan
Department of Electrical and Computer Engineering
University of Alberta,
Edmonton T6G 2V4, Canada
lkurgan@ece.ualberta.ca

Abstract

Fuzzy cognitive maps (FCM) are a tool that is used for analyzing and aiding decision making. They incorporate elements of fuzzy logic and neural networks. FCMs are applied in problems involving complex webs of casual relationships, which often include feedback, and where qualitative rather than quantitative measures of influences are available. They are very easy to understand, use, and analyze. On the other hand, they require domain expertise during the design process.

This paper presents application of FCMs in the software engineering domain. Two models, which concern software development, are proposed and discussed. They are used to describe and analyze factors, which affect pace of work progress during software project. The performed simulations show very interesting relationships that agree with the theoretical results reported in the literature.

Keywords

Fuzzy Cognitive Maps, Management of Software Project, Project Communication.

1. Introduction

Project management is a technique for matching available resources (time, money, and people) against business project aims (early completion date and final cost) [10]. Products of software projects have some features, such as invisibility, complexity and flexibility, which makes software project different from projects in other disciplines. Some methods of general project management are not valid to software project management [5]. According to statistical analyses, the most frequent challenge to software project manager is coping with keeping deadlines [18]. Origin of this phenomenon is connected to difficulties in estimation of software projects. What is worse, the progress of software projects is also hard to foreseen. As a result, many projects exceed time and budget, e.g. the £339 million United Kingdom air traffic control system was reported as being two years behind schedule [3]. Objective of this paper is to analyze factors, which affect progress in software project. The paper is concentrated on effects that result from communication among workers during software projects. This issue plays a very important role, which is often underestimated. Additionally, an evaluation of effects after adding new people to a project in order to complete it faster is also performed.

Software development requires collaboration among people. Many people who have different backgrounds, such as domain experts, analysts, designers, programmers, managers, technical writers, graphic designers, and users, need to be involved in a software development process to make it successful [14]. It is essential to share information among these people in an accurate and timely manner. Communication forms and types are widely described in literature [1], [14]. It is commonly known that communication is an inseparable element of software development project, and it significantly affects progress of the development. The range of this impact varies and is connected with ability be partition the project. This paper examines the above relationships by analyzing project progress with respect to the number of people involved in it and communication among the project's participants.

Many software projects are not successful due to a lack of time to complete them. Brooks lists five essential elements which contribute to such troubling situations [1]. One of them is incorrect reaction to the situation when schedule slippage occurs. The most common response when a project is behind schedule, which is adding additional manpower, is often ineffective. What is worse, it sometimes causes even greater delay according to the Brook's Law that states that "…adding manpower to a late software project makes it later…" [1]. Software managers usually act on the basis of an assumption that increase in number of workers involved in project is the best solution to finish it on time. Such reasoning steams from fallacious thought that progress can be expressed in the man-month unit of effort. Unfortunately this is not true. This measure is good for describing cost of a software development project, but it is not suitable for describing the progress. Brooks reports that the length of a project depends on its sequential constraints and the maximum number of men that develop the software depends on number of independent subtasks [1]. Thus, it is vital to establish these two quantities to properly schedule project. It is very important to understand that exceeding maximum number of workers involved in the project may result in decreasing the relative progress. This results from costs associated with both training people and repartitioning of the tasks, which are necessary when new workers are added to help finishing project faster. This paper aims to verify the hypotheses stated above.

There are many techniques used to model dynamic systems. In general, they can be divided into two major groups [11]. The first one concerns quantitative techniques. They can be applied both to well-understood systems, such as mathematical programming techniques of operation research, and to less-understood ones, such as statistically based methods of data mining. However, significant effort and specialized knowledge outside the domain of interest is required to develop these models. What is more, some dynamic systems can be nonlinear, which sometimes makes impossible to use the quantitative models. The modeling techniques from the second group concern qualitative approach. They allow developing models on the basis of knowledge in the domain of application, and often incorporate feedback mechanism. Since the relationships that concern software development project fall into the second group, the Fuzzy Cognitive Maps, which are one of the quantitative models, were used to perform simulations.

2. Fuzzy Cognitive Maps

Cognitive Map is a tool, which can be used for modeling and simulation of complex systems [7], [8]. It allows simulating behaviour of black box systems through use of cause and effect relationships. They consist of a collection of nodes linked by edges, which are used to describe a given system. The nodes represent concepts, or variables, relevant to a given domain, whereas the causal links between them are represented by edges. The edges are directed to indicate the direction of causal relationships. Additionally, each edge includes information on type of the relationship. The relationships can be positive (a promoting effect) or negative (an inhibitory effect). The drawback of cognitive maps is that they do not allow feedback, which significantly limits its usefulness.

Fuzzy Cognitive Maps are an extension of Cognitive Maps [7], [8]. Two significant improvements introduced in the FCMs are that:

1. Causal relationships between nodes are fuzzified. This feature enriches description of connection by numerical value instead of only using signs. It allows using varying degrees of causal influence.
2. The system is dynamic, which means that it evolves with time, and involves feedback mechanisms. Specifically, the effect of change in a concept node may affect other concept nodes, which eventually can affect back the node that initiated the change.

The strength of relationship between two nodes usually takes on any value in the [-1, 1] range [7], [8]. Value -1 represents full negative, whereas +1 full positive causal effect. Zero denotes no causal effect. Other values correspond to different fuzzy levels of causal effect. Definition of system relationships can be described by a matrix, called connection matrix. Considering the system with *n* concept nodes, we have *n* by *n* matrix, which elements are the causal link strengths.

State of the system is determined by a state vector, denoted C_k, which specifies current values of all system variables, which can be understood as a kth state of the system. The FCMs iteratively update the state of the system. During each iteration value of each node is calculated based on the current values of every node (concept), which exerts influence on it through a causal link. After multiplying these values by edge weight between the two nodes, which represents the strength of the relationship between the nodes, the sum of these products is taken as the input to a transformation function. The function is used to reduce unbounded inputs to a certain range. This function usually generates either binary, when threshold function is used, or continuous, when sigmoid squashing function is used, concept values.

The value of each node in any iteration is computed from values of nodes in preceding state, using approach shown in Figure 1 and the following equation [7]:

$$N_j(k+1) = f\left(\sum N_i(k) \cdot e_{ij}\right)$$

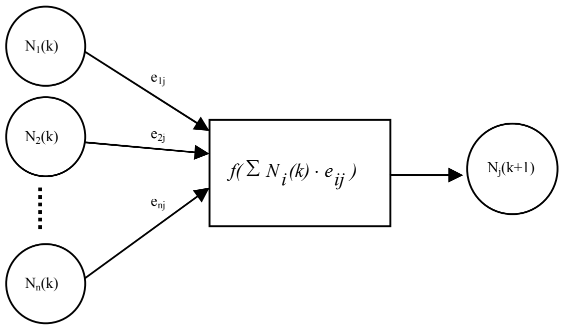

Figure 1. Computation of a new concept node's value.

where $N_i(k)$ is the value of ith node in the kth iteration (system state), e_{ij} is edge weight between nodes N_i and N_j, and f is the transformation function.

Three types of transformation function which are commonly used are shown below [11]:

1. Bivalent

$$f(x) = \begin{cases} 0, & x \le 0 \\ 1, & x > 0 \end{cases}$$

2. Trivalent

$$f(x) = \begin{cases} -1, & x \le -0.5 \\ 0, & -0.5 < x < 0.5 \\ 1, & x \ge 0.5 \end{cases}$$

3. Logistic signal

$$f(x) = \frac{1}{1+e^{-cx}}$$

The last function is a continuous-output function. The constant c is critical in determining the degree of fuzzification of the function.

The possible results of a simulation performed with FCM depend on transformation function [7]. Using function which results in binary values, the simulation of a FCM system heads for either fixed pattern of node values, which are called *hidden pattern* or *fixed-point attractor*, or keeps cycling between a number of fixed states, which are known as the *limit cycle*. Using a continuous-output transformation function may result in a different outcome. System may continue to produce different state vector values for successive cycles. This unstable situation is called *chaotic attractor*.

FCMs have been applied in many different areas. Examples include modeling of plant control [4], modeling of political affairs is South Africa in the apartheid era [9], analysis of electrical circuits [15], disease diagnosis [17], modeling of virtual worlds [2], analysis of failure modes effects [13], fault management in distributed network environment [12], modeling and analysis of business performance indicators [6], and modeling of supervisors [16].

3. Experiments

The paper proposes and simulates two models, which concern management of a software development project.

The first model describes initial development phase. It characterizes a situation when a team of workers starts to do a software development project. Successive states of the modeled system show changes in considered nodes, which represent software management aspects. Values of the nodes describe trends, i.e. qualitative changes evolving with time, of the considered aspects. The final state achieved by the model shows the development state where equilibrium between the considered concepts, with respect to the initial human resources, is achieved. The final state shows the relative development progress speed that is achieved for the modeled system.

The second model extends the first model. It is used to analyze what happens in case when any corrections to the equilibrium state need to be performed. In particular, it allows simulating influence of adding new workers to a project that is already in progress.

The experiments was carried out using the logistic signal function as a threshold function, which is a continuous-output transformation function and thus provides true fuzzy conceptual node states. The constant c was set to 5.

3.1. First Model

The first model presents relations, which are essential during software development project. It consists of three concept nodes, and is shown in Figure 2.

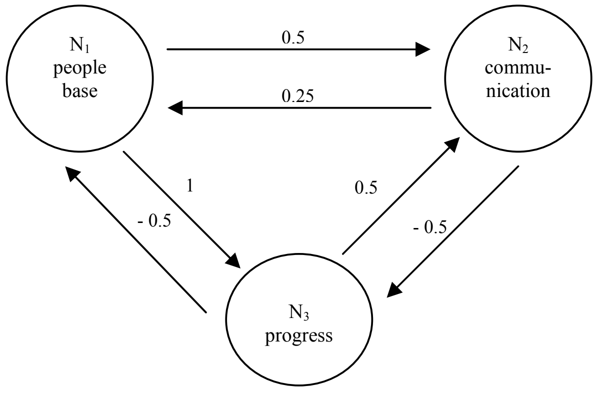

Figure 2. FCM describing software development project.

The following nodes are their representations are used:

- *people base* (N_1) – people involved in project, which includes both designers and implementers.
- *communication* (N_2) – communication effort, which reflects effort connected with cooperation among people working on a project.
- *progress* (N_3) – development abilities, which can be interpreted as a factor, which describes rapidity of the development of the project.

The casual relationships between nodes, which are represented by directed edges, can be interpreted as follows. Increase in number of people who are assigned to work has positive effect on progress (+1 directed edge between N_1 and N_3). However, the effect is not as simple as it looks at a first glance. This forces higher level of communication among workers (+0.5 directed edge between N_1 and N_2), which is essential to ensure proper partitioning of tasks and cohesion of their work. Furthermore, this negatively influences the progress (-0.5 directed edge between N_2 and N_3). On the other hand, increase in progress can lead to limiting number of people involved in project, e.g. they can be transferred to develop other projects (-0.5 directed edge between N_3 and N_1). However, higher value of progress also implies increase in communication (+0.5 directed edge between N_3 and N_2). This, in turn, can cause a trend to increase number of workers (+0.25 directed edge between N_2 and N_1). The strength of the relationships was established experimentally, and reflects common perception of the strength of these relationships.

3.1.1. Simulation

Next, the developed model was simulated. The starting vector is denoted C_0. Each state vector consists of three numbers, which correspond to conceptual nodes as follows: people base (N_1), communication (N_2), and progress (N_3). The experiment makes possible to examine the mutual relationships among these elements. The simulation begins with following start state vector $C_0 = (0.5, 0, 0)$, which represent a situation when people base concept is active and set at value 0.5, and other concepts are inactive.
This state can be interpreted as the beginning stage of software development project. Software project manager assigns some people to work on a given project. Communication and progress nodes are inactive, i.e. their values are set as zero, what indicates that the workers did not start to work yet.

As the simulation continues successive values of nodes show trends which occur with the progressing time. By analyzing states of nodes in consequent system states, relationships between the nodes can be learned and analyzed.

Rounding to three significant digits, during the simulation the following states are achieved:
$C_0 = (0.500, 0.000, 0.000)$
$C_1 = (0.500, 0.777, 0.924)$
$C_2 = (0.208, 0.972, 0.636)$
$C_3 = (0.408, 0.892, 0.199)$
$C_4 = (0.650, 0.820, 0.452)$
$C_5 = (0.474, 0.940, 0.768)$
$C_6 = (0.322, 0.957, 0.504)$
$C_7 = (0.484, 0.888, 0.314)$
where C_i is the ith state of the system.
Next, the model steadily reaches equilibrium, which is state $C_{79} = (0.469, 0.920, 0.511)$.

3.1.2. Analysis of the Simulation

In order to better understanding achieved results, values of all nodes for first 20 states are presented in Figure 3.

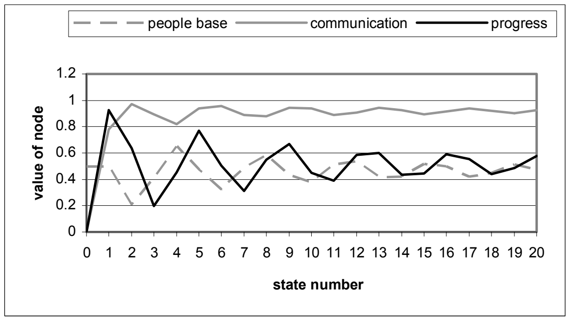

Figure 3. Results of the simulation of the first model.

The first state (C_1) shows that progress node achieves the highest value. This reflects situation when tasks have been partitioned and assigned to workers. In this phase they can start their work separately, so the progress rate is high. However communication problems occur and increase rapidly, which has strong negative impact on progress. It can be interpreted as actions taken by a manager in order to ensure the cohesion, or another words intellectual integrity of the project [1]. In general, as described in the introduction, communication effort is present during all phases of development of software project.

The next states show very interesting trends. Communication remains at high level all the time and, along with people base, they influence progress, which significantly oscillates during the several initial states. The oscillations are mainly caused by adjustments in the values of people base, which in turn are performed in reaction to the amount of progress. This situation can cause problems in proper estimation of project effort, and therefore will have impact on scheduling. The simulation ends in reaching an equilibrium state.

Comparing the state C_1 with the equilibrium state, one can see that value of people base node remains approximately at the same level, yet initial progress node value decreases almost twice. This means that mutual relationship between these two factors is not linear, and therefore it is not possible to easily control progress by means of number of people assigned to project. The analysis shows that the manager must wait some time before making any judgments based on the progress to accommodate for the communication between the team workers.

3.2. Second Model

The second model describes situation when new people are added to help completing project faster. This results from a situation when a prognosis based on current progress shows that the project's deadline will not be met, and therefore a corrective action needs to be performed. The FCM proposed to model this situation is presented in Figure 4

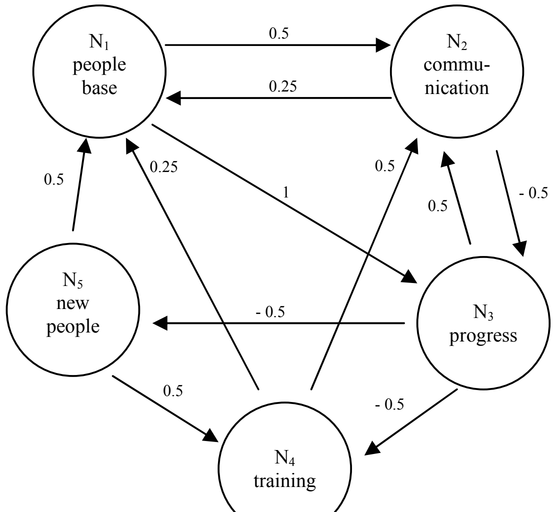

Figure 4. FCM describing adding new workers to an existing software development project.

The following nodes are their representations were added with to the first model to build the second model:

- *training* (N_4) – effort connected with introducing new people to project
- *new people* (N_5) – workers added to project, which can be interpreted as a factor, which describes strength of a trend of adding new personnel to the project

Meaning of the other nodes remains the same.

This model introduces both new concepts nodes and new casual relationships among them. The relationships, which are represented by directed edges between nodes, can be interpreted as follows. By adding new workers the people base increases (+0.5 directed edge between N_5 and N_1). However, this process does not influence progress directly, because they have to be trained first (+0.5 directed edge between N_5 and N_4). Training requires communication among people, so the positive connection between these nodes is present (+0.5 directed edge between N_4 and N_2). It also influences increase in people base (+0.25 directed edge between N_4 and N_1). On the other hand, the progress negatively influences both adding new people (-0.5 directed edge between N_3 and N_5) and training (-0.5 directed edge between N_3 and N_4).

3.2.1. Simulation

This time the simulation is started with a vector, which describes equilibrium state from the first model. This is because the second model concern the same project as the first model, where an equilibrium state has been achieved, i.e. the project is set according initial human resources. The second model shows what happens when the existing pace of software development project is insufficient. A common reaction to this situation is to add new people to increase development speed. The model reflects this situation, and therefore the simulation starts with the following state:

$C_0 = (0.469, 0.920, 0.511, 0.000, 0.500)$

In this case, each state vector consists of five numbers, which correspond with nodes values as follows: people base, communication, progress, training, and new people. The initial state vector C_0 uses three initial values from the first model, the value of new people node is set on 0.5, and training is set to be inactive, i.e. with value of zero. This describes circumstances just after new workers are added to project.

Rounding to three significant digits, the simulation of the model results in obtaining the following states:

$C_0 = (0.469, 0.920, 0.511, 0.000, 0.500)$
$C_1 = (0.917, 0.921, 0.511, 0.493, 0.218)$
$C_2 = (0.910, 0.992, 0.907, 0.324, 0.218)$
$C_3 = (0.899, 0.995, 0.888, 0.151, 0.094)$
$C_4 = (0.841, 0.992, 0.882, 0.121, 0.098)$
$C_5 = (0.837, 0.990, 0.849, 0.124, 0.099)$
$C_6 = (0.838, 0.989, 0.847, 0.133, 0.107)$
$C_7 = (0.842, 0.989, 0.847, 0.136, 0.107)$
where C_i is the i^{th} state of the system.

Next, the model steadily reaches equilibrium, which is state $C_{12} = (0.842, 0.990, 0.850, 0.135, 0.107)$.

3.2.2. Analysis of the Simulation

In order to help analyzing the results, the node values are shown in Figure 5.

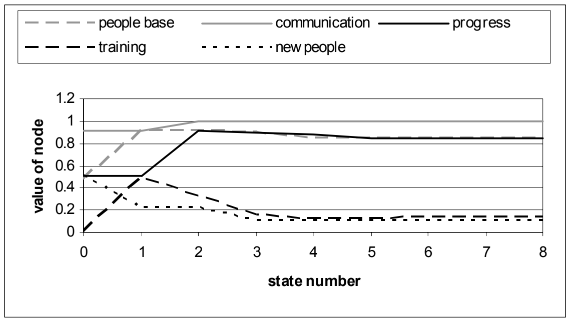

Figure 5. Results of the simulation of the second model.

The expected result in this simulation is to observe rapid increase in progress upon adding new people to the team. The simulation shows that this effect does not come immediately. The new workers have to be introduced to project and tasks must be repartitioned. This effort limits their usefulness to improving progress, resulting in progress remaining unchanged between the initial and the first state, i.e. transition between C_0 and C_1.

At the state C_1, the manager faces the situation when increase of cost of project development connected with paying new workers has happened, but the expected boost in progress does not come. This is a very dangerous state when the manager can be tempted to add even more people to the project, which results in so called regenerative schedule disaster [1]. It can be seen that instead of progress, the training efforts rise rapidly. In the states following the second state, an increase in progress can be observed, since the new staff is trained and prepared for work.

Focusing our attention on progress node, some interesting observations can be made. After achieving its maximum value in state C_3, progress value starts to decline. This trend remains unchanged until the model reaches the equilibrium state. It is caused by diminishing values of people base and achieving the maximum value by communication node.

Two conclusions can be drawn from analysis of the simulation. First, the manager again must wait some time before making any judgments based on the progress to accommodate for the training of the new team workers. (S)he should not add additional manpower too early since this may result in additional costs in terms of training and communication, which can even result in decreasing the progress. Second, adding new workers, when little time is left to finish the project is not a good idea, since initially the costs in introducing new people may be higher that the gains.

3.3. Other Simulations

Several additional simulations were also performed with the second model. This time, the initial state vector was changed and the influence of these changes on the behaviour of the entire system was studied. Several experiments, which illustrate how the initial value of the new people node impacts on the progress, were performed. Initial values of the other nodes remained unchanged, which allows observing and comparison of influence of varying initial values of the new people node on the system behaviour. The obtained results are summarized in Figures 6, 7, and 8.

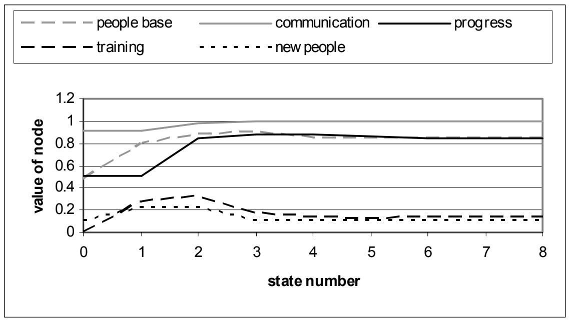

Figure 6. Simulation results for initial value of new people set to 0.1.

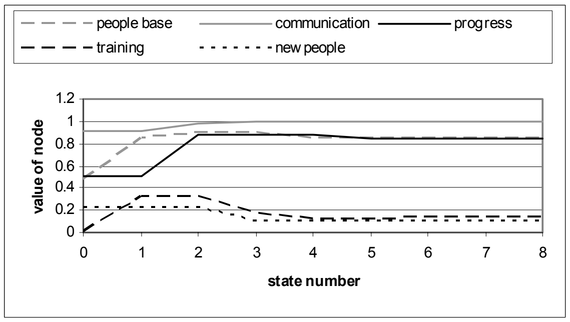

Figure 7. Simulation results for initial value of new people set to 0.22.

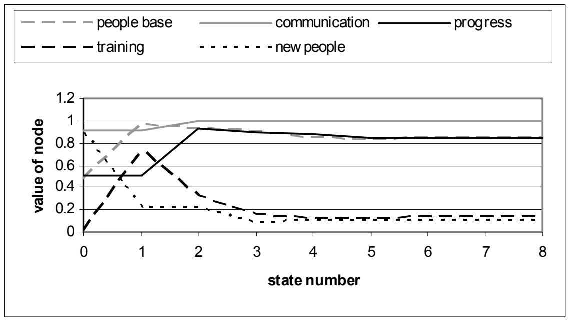

Figure 8. Simulation results for initial value of new people set to 0.9.

The analysis of the above results shows that the final progress value does not change, no matter what is the level of the initial state of the new people node. However, these results differ from each other significantly in the value of training node in the first state of the simulation. In general, the higher is the value of this node the greater the cost of project. It is important to notice that the training and communication forces to trim the amount of newly introduced people to the same levels despite different initial costs. Therefore, the effort that has been spent on training new people in the last simulation (state C_0) is unjustified. What is worse, the progress in the last simulation tends to decrease very little over time. This can lead to difficulties in creating proper project schedule, if higher value of progress than in fact it is would be presumed. All these results confirm the observation noted in the introduction, that duration of software project cannot be controlled by simply adding new workers. The other factors, like communication, training, and susceptibility of a given project to partitioning, have to be taken into consideration. If one will not accommodate for these factors, the expected results will substantially differ from his or her expectations.

4. Summary and Conclusions

FCMs are convenient to simulate systems when relationships are not easy to describe by mathematical formulas. Problems that are considered in this paper belong to this category. FCMs allow building and simulating models just on the basis of knowledge about mutual relationships between factors which are present in a given problem.

This paper describes development and analysis of two models that were used to observe influence of factors, such as communication and training, on the pace of the software development project. Literature indicates that communication efforts are often underestimated, which often leads to missed project deadlines.

The experiment, which was carried out using one of the models, confirms that communication has a great effect on the progress of software development project. Neglecting it can lead to false assumptions concerning project schedule and, in consequence, cause exceeding its planned time. The manager must wait some time before making any judgments to be able to observe a true progress value that accommodates the cost do this factor.

The other model was used to describe situation that often occurs when software project is behind schedule. In this case, the usual response is to add new people to the project. The obtained results agree with those found in the literature. Namely, adding new people should be performed very carefully, because initially it does not bring wanted effects, but rather leads to increase of the costs. Here also the manager must wait some time before making any judgments based on the current progress to accommodate for the training of the new workers. Also, adding new people to project that are soon to be finished should not be pursued since again the initial costs may be higher that the future gains. What is more, the model shows that a larger value of progress cannot be gained by just increasing the number of new workers, which is mainly caused by the cost of both training of new workers and the cost of increased communication.

In the nutshell, proper estimation of software development project before its beginning is of great importance, and of great difficulty. One needs to accommodate for many factors, like communication and training when performing the estimation. This paper shows that tools, such as FCMs, can provide valuable helps in both understanding and modeling relationships that can improve accuracy of the estimates.

5. References

[1] Brooks, F., *The Mythical Man-Month: Essays on Software Engineering*, Anniversary Edition, Addison-Wesley, 1995.

[2] Dickerson, J., and Kosko, B., Fuzzy Virtual Worlds, *Artificial Intelligence Expert*, vol.7, pp.25-31, 1994.

[3] Flower, S., *Software Failure, Management Failure,* Wiley & Sons, 1996.

[4] Gotoh, K., Murakami, J., Yamaguchi, T. and Yamanaka, Y, Application of Fuzzy Cognitive Maps to Supporting for Plant Control, *Proceedings of the SICE Joint Symposium of Fifteenth Systems Symposium and Tenth Knowledge Engineering Symposium*, pp.99-104, 1989.

[5] Hughes B., and Cotterell M., *Software Project Management*, McGraw-Hill, 1999.

[6] Kardaras D., Mentzas G., Using Fuzzy Cognitive Maps to Model and Analyze Business Performance Assessment, In *Advances in Industrial Engineering Applications and Practice II*, Chen, J., and Mital, A., (Eds), pp.63-68, 1997.

[7] Khan, M., and Quaddus, M., Fuzzy Cognitive Map as a Tool for Group Decision Support, *Proceedings of the 2002 Group Decision and Negotiation Conference*, 2002.

[8] Kosko, B., Fuzzy Cognitive Maps, *International Journal of Man-Machine Studies*, vol.24, issue 1, pp.65-75, 1986.

[9] Kosko, B., *Neural Networks an Fuzzy Systems*, Prentice-Hall, 1992.

[10] Microsoft Encarta Encyclopedia, 1993-2002 Microsoft Corporation, 2003.

[11] Mohr, S., *The Use and Interpretation of Fuzzy Cognitive Maps,* Master's Project, Rensselaer Polytechnic Institute, 1997.

[12] Ndousse, T., and T. Okuda, Computational Intelligence for Distributed Fault Management in Networks Using Fuzzy Cognitive Maps, *Proceedings of the IEEE International Conference on Communications Converging Technologies for Tomorrow's Application*, pp.1558-1562, 1996.

[13] Pelaez, C.E. and J.B. Bowles. 1995. Applying Fuzzy Cognitive Maps Knowledge Representation to Failure Modes Effects Analysis, *Proceedings of the IEEE Annual Symposium on Reliability and Maintainability*, pp.450-456, 1995.

[14] Project Communication, draft, available at http://citeseer. nj.nec.com/323780.html, 1999.

[15] Styblinski, M., and Meyer, B., Signal Flow Graphs versus Fuzzy Cognitive Maps in Application to Qualitative Circuit Analysis, *International Journal of Man-Machine Studies*, vol.35, pp.175-186, 1991.

[16] Stylios, C., and Groumpos, P., the Challenge of Modeling Supervisory Systems using Fuzzy Cognitive Maps, *Journal of Intelligent Manufacturing*, vol.9, issue 4, pp.339-345, 1998.

[17] Taber, R., Knowledge Processing with Fuzzy Cognitive Maps, *Expert Systems with Applications*, vol.2, pp.83-87, 1991.

[18] Thamhain, H., Wilemon, D., Criteria for Controlling Software According to Plan, *Project Management Journal*, pp.75-81, 1986.