Parallel Fuzzy Cognitive Maps as a Tool for Modeling Software Development Projects

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

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

W. Pedrycz
Department of Electrical and Computer Engineering
University of Alberta
Edmonton, Alberta T6G 2V4, Canada
pedrycz@ee.ualberta.ca

M. Reformat
Department of Electrical and Computer Engineering
University of Alberta
Edmonton, Alberta T6G 2V4, Canada
reform@ee.ualberta.ca

Abstract - **Fuzzy cognitive maps (FCM) are useful tool for simulating and analyzing dynamic systems. The FCMs have a very simple structure, and thus are very easy to comprehend and use. Despite of the simplicity, they have been successfully adopted in many different areas, such as electrical engineering, medicine, political science, international relations, military science, history, supervisory systems, etc.**

Software development is a complex process, and there are many factors that influence its progress. To effectively handle larger development processes, they are usually divided into subtasks, which are assigned to different teams of workers, and often are performed in parallel. However, some constraints that impose particular sequence of realization of these subtasks, i.e. some tasks cannot be started before completing others, usually exist. Proper division of a project into subtasks and establishing relations between them are essential to correctly manage software projects. Neglecting these constraints often leads to problems that, in consequence, cause misestimating the overall time and budget.

This paper introduces a new architecture of FCM, which combines a number of simple FCM models that work simultaneously into a novel parallel FCMs model. It uses a special purpose coordinator module to synchronize simulation of each FCM model. This approach extends application of FCMs to complex systems, which contain multiple subtasks that run in parallel, and thus must be simulated with multiple FCM models. In addition, application of parallel FCMs to analyze and design software development processes is presented. FCM models are focused on simulating and analyzing factors, such as progress and communication, and their relationships, which are based on theoretical research studies and practical implementations. The parallel FCM model is used to simulate complex projects where multiple tasks exist. The paper is based on our previous work where FCM models, which describe relationships between the above factors for individual development tasks, were developed. The newly proposed architecture allows for efficient analysis of dependences between tasks performed in parallel.

KEYWORDS

Parallel Fuzzy Cognitive Maps, Management of Software Project, Gantt Chart, Software Development Project

I. INTRODUCTION

Fuzzy Cognitive Maps are a tool that is used for modeling and simulation of dynamic systems. The FCMs have a very simple structure, in terms of a graph that consists of nodes and directed edges, and a very simple execution model that allows for fast dynamic simulations. They have been applied in many different areas, such as disease diagnosis [18], analysis of electrical circuits [15], analysis of failure modes effects [13], fault management in distributed network environment [12], modeling and analysis of business performance indicators [10], modeling of supervisors [16], modeling of software development project [14], modeling of plant control [6], modeling of political affairs in South Africa [11], and modeling of virtual worlds [4].

The approach using a single FCM model is suitable to describe various dynamic systems, as presented in the previous paragraph. However, some complex dynamic systems cannot be simply modeled with a single FCM, especially in case when the system consists of multiple, parallel processes. In order to apply FCMs theory for such systems, a special architecture, called parallel FCM, is proposed. In this architecture, multiple FCMs communicate with a coordinator module, which control overall simulation and simulation of each model according to given constraints and rules.

This paper presents application of parallel FCMs in the software engineering domain. Software projects are excellent example of a system where multiple tasks are performed simultaneously. Here, each task is described by a FCM model, which shows relationships between factors that affect progress of work. The design of individual FCM models is based on theoretical research presented in [3] and extends model introduced in [14]. The structure of parallel FCM allows executing the individual models (software tasks) in parallel, and in accordance with a given scenario.

Software project management is the “process of planning, organizing, staffing, monitoring, controlling and leading a software project” [9]. Products of software projects have some unique features, like invisibility, complexity and flexibility, which cause difficulties in managing projects in accurate manner [7]. There are, essentially, three project estimation approaches: *human-based, algorithmic*, and *machine learner-based* [2]. However, it is commonly known fact that estimation of software project is very difficult. Surveys clearly indicate that many project failed due to lack of time [5]. The extent and impact of this problem is quite substantial. According to a survey performed by the Standish Group in mid 1990's, which included over 8,000 software projects, an average project exceeded its planned budget by 90% and its schedule by 222% [16]. Also, more than 50% of the completed projects implemented less than 50 percent of original requirements [16].

Origin of this phenomenon is connected with complexity of factors that affect progress of work during development of software projects. Some researchers estimate software project progress based on very simple machine learning models, which were used to find relationship between the sources lines of code for each of the envisioned components in the end product and a project programming effort [1]. This paper is concentrated on analyzing issues that are associated with communication among workers. We note that underestimating this factor often leads to expensive project management errors. The experiments performed in [14] confirmed that communication effort is an essential part of software projects, which has to be taken into consideration in order to accurately manage projects.

Software project is a collection of inter-related tasks. In general, there are three approaches to identifying tasks that are present in software project – *activity-based approach, product-based approach*, and *hybrid-approach* [7]. With respect to the start time constraints, tasks can be divided into two types. The first group includes tasks that can be begun without any starting conditions, whereas tasks belonging to the second group require that others have to be completed before they can begin (it is commonly known as *precedence requirements*). In order to properly manage a project it is essential to have a plan, which must include the start and completion times for each task [7].

What adds to the complexity of the management of software projects is the fact that there are several different types of software development tasks, depending on their nature. Brooks lists four tasks types, i.e. perfectly partitionable task, unpartitionable task, partitionable task requiring communication, and task with complex interrelationships [3]. Many software project managers act on the basis of assumption that all tasks belong to the first category, which would allow them to control project progress by number of involved people. Unfortunately, this group of tasks practically does not exist in real projects. This implies that progress cannot be expressed just in the man-month unit of effort. The length of a project is connected with its sequential constraints, and the maximum number of workers depends on independent subtasks [3]. This paper extends approach described in [14] by introducing new FCM-based models that describe each of the above task types.

As progress of software project development goes on, it is important to collect and present the data about this process. One of the methods, which are used to track project progress, is the Gantt chart [7]. This is a task bar chart that indicates scheduled task dates and durations. Gantt chart is constructed with a horizontal axis that represents the total span of the project and a vertical one that represents the tasks that constitute the project. The Gantt chart also includes progress indicators (e.g. as shaded tasks bars) and a 'today cursor'. All these items allow assessing which tasks are ahead or behind the schedule.

II. Fuzzy cognitive maps models

The paper exploits and extends a FCM model that consists of three concept nodes that show relationships during software project development [14]. However, several important changes, which concern both concepts understanding and relationships strength, were introduced. The new base model is shown in Fig. 1.

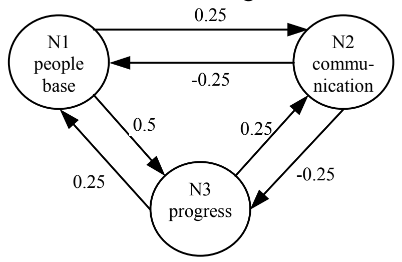

Fig. 1 FCM describing software development project

The nodes represent the following concepts:

- people base (N_1) – degree of exploitation of people involved in project with respect to the initial human resources, which includes both designers and implementers, i.e. effectiveness of their work;
- communication (N_2) – communication effort, which reflects effort connected with cooperation among people working on project;
- progress (N_3) – development abilities, which can be interpreted as a factor, which describes rapidity of the development of the project.

Following, the interpretation of causal relationships between nodes is explained. Increase in people's effectiveness has positive effect on progress (+0.5 directed edge between N_1 and N_3). However, it forces higher level of communication among workers (+0.25 directed edge between N_1 and N_2), which is critical to ensure cohesion of their work. Moreover this has negative impact on the progress (-0.25 directed edge between N_2 and N_3). On the other hand, increase in progress can have a positive impact on effectiveness through factors, such as motivation (+0.25 directed edge between N_3 and N_1). What is more, higher value of progress requires higher level of communication (+0.25 directed edge between N_3 and N_2). This, in consequence, increased progress can cause decrease in effectiveness of work (-0.25 directed edge between N_2 and N_1). The strength of the relationships was established in the following way. First, the influence of a concept on another between each pair of concepts was determined as "negative" or “positive”. Then, these relationships were expressed in fuzzy terms, i.e. *weak, medium, strong* and *very strong* by taking into consideration the common perception of their strength. Finally, these terms were replaced by the numerical values 0.25, 0.5, 0.75 and 1, respectively.

In this paper, an extension of the above model is presented. Three models, that include the same concepts, but describing different types of tasks, as defined in [3], are proposed. Below, a brief description of each task type is shown along the proposed FCM model. A plot that is a result of the model simulation, and shows number of iterations, which are necessary to complete task of given complexity (it was arbitrarily set at 100) as a function of initial people base node value, is also shown.

A. *Unpartitionable task* – this task cannot be partitioned because of sequential constraints. As a result adding more manpower has no effect on the schedule, see Fig. 2 and Fig. 3.

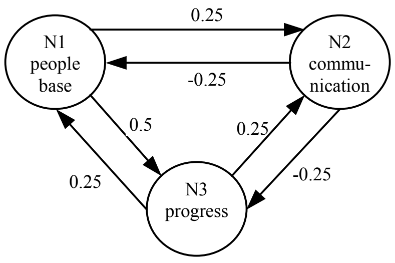

Fig. 2 FCM model of an unpartitionable task

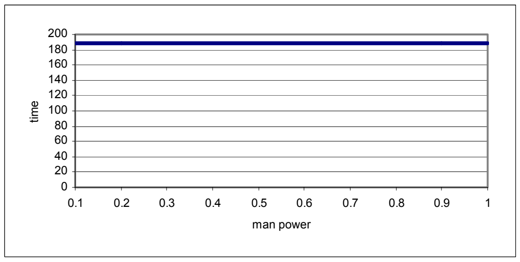

Fig. 3 Time versus number of workers - unpartitionable task

B. *Partitionable task requiring communication* – this kind of tasks can be partitioned. However, this process requires communications among people and the effort of communication must be added to the amount of work to be done.

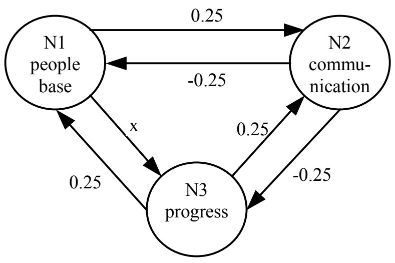

Fig. 4 FCM model of a partitionable task requiring communication

In order to control the progress rate by number of people involved in it, one of the edge values of model, presented in Fig. 4, is parameterized. The variable *x* reflects the impact on progress by effectiveness of work. This value is fixed before starting simulation with an initial value of people base node. The model represents the situation where the more workers are assigned to the task the higher is the progress, but with the dumping effect represented by the communication node, see Fig. 5.

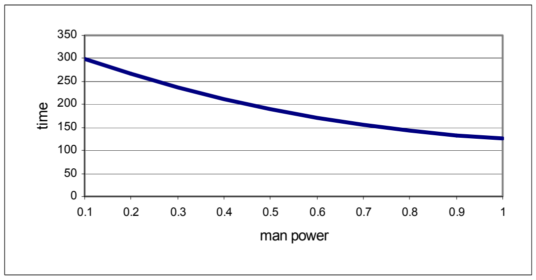

Fig. 5 Time versus number of workers - partitionable task requiring communication

C. *Task with complex interrelationships* – in some cases, the communication effort may fully counteract the progress of the work when too many people are added. This problem occurs when each part of task must be separately coordinated with each other part.

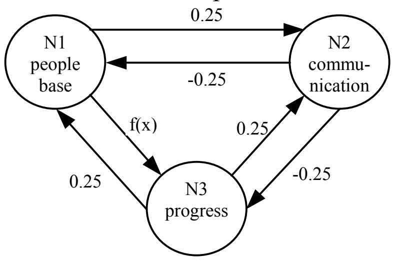

Fig. 6 FCM model of a task with complex interrelationships

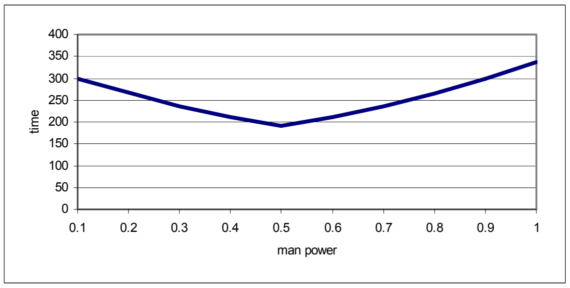

Fig. 7 Time versus number of workers - task with complex relationships

The FCM structure, see Fig. 6, is identical to the structure shown in Fig. 4. However, this time the modified edge value is given as a function of initial value of people base node. The function type was selected experimentally to obtain model that fit the behavior of this type of task.

$$f(x) = -|x - 0.5| + 0.5$$

where x is the initial people base node value.

Value of this function is established prior to start the simulation and given as a weight of directed edge between N_1 and N_3. Therefore all model weights values are fixed before starting simulation, and do not change during its execution.

III. Parallel Fuzzy Cognitive Maps

The paper introduces a novel parallel FCM architecture. The architecture coordinates multiple FCMs into a single model that is able to simulate and coordinate the individual FCM models that are executed in parallel, and produce the overall simulation results. The architecture is shown in Fig. 8.

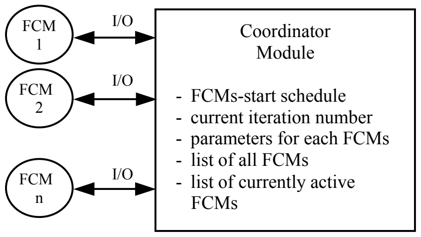
FCM 1
I/O
FCM 2
I/O
FCM n
I/O
Coordinator Module
- FCMs-start schedule
- current iteration number
- parameters for each FCMs
- list of all FCMs
- list of currently active FCMs

Fig. 8 Parallel Fuzzy Cognitive Maps – architecture of the system

The *coordinator module* is responsible for triggering FCMs that are included in the system, coordinating the individual FCMs, as well as checking when the simulation should stop. The *I/O* enables to send information between individual FCMs and the coordinator module. The system can trigger, stop, or change parameters of the individual FCMs through the I/O. Each individual FCM sends values of its all concept nodes to the coordinator module in each iteration of the execution. The coordinator module is able to simulate the parallel FCM by analyzing and coordinating current state of each individual FCM, and setting their initial parameters.

The execution model of the parallel FCM in given in the form of the following pseudo-code:

```
A. Initialize the parallel FCM:
    1. Create individual $FCM_i$, $i=1, 2, \ldots n$;
       Each $FCM_i$ is initialized with predefined structure and
       weights values. Each $FCM_i$ had defined a stopping
       condition that is used to terminate its execution.
    2. Initialize the list of all individual $FCM_i$.
       $AllFCM=\{FCM_1, FCM_2, \ldots, FCM_n\}$;
    3. Initialize the FCM-start schedule.
       The schedule lists the start time for each $FCM_i$ in terms
       of a fixed time (iteration) or dependency on finishing
       set of preceding other individual FCMs $\{FCM_j\}$, where
       each $FCM_j$ starts before $FCM_i$, and $j \neq i$;
    4. Initialize the list of active individual $FCM_i$.
       $ActiveFCM=\{\}$;
B. At each iteration of the parallel FCM:
    1. Check which tasks can be started according to the
       FCM-start schedule, and add them into the
       ActiveFCM;
    2. Perform one iteration for each FCM from ActiveFCM;
    3. Check which tasks from ActiveFCM are finished and
       remove them from AllFCM;
    4. If $AllFCM=\{\}$ then TERMINATE, otherwise go to B.1.
```

Each individual FCM model is described by set of parameters (weights and structure) that are used by the coordinator module to perform the simulation. Each individual FCM has also defined a stopping condition that terminates its execution according to a predefined criterion, e.g. achieving stable state, max number of iterations, etc. These parameters are established in step A.1. The individual FCM models are executed in parallel according to a schedule given by the user, which is initialized in step A.3.

In case of the application of the parallel FCM to analysis of software development projects, each individual FCM represents a given development task. The tasks are combined using schedule, which is usually represented by a Gantt chart. The simulation of the parallel FCM model enables to analyze relationships between specific tasks, and their impact on the overall schedule. The initialization parameters are computed by analyzing project description and Gantt chart:

A. Structure and weights of each individual FCM_i are defined by selecting one of three types of FCM model, which are described in section II. The selection is performed based on the work type associated with a given development task.

B. Each individual FCM_i has defined a stopping criterion that is used to stop its simulation. This criterion is defined in terms of total amount of work that needs to be performed in the task modelled by a given FCM_i. The value is derived from the length of the task defined in the Gantt chart.

C. Schedule is derived from the Gantt chart. Each individual FCM_i will be started either in a specified time (iteration), or must wait until others FCM_i are completed, as given in the chart.

IV. Experiments

The objective of these experiments is to assess suitability of proposed parallel FCM structure to simulate scheduling of software project. The goal of the simulations is to allow examining the influence of different initial conditions on the anticipated schedule, as defined by the Gantt chart.

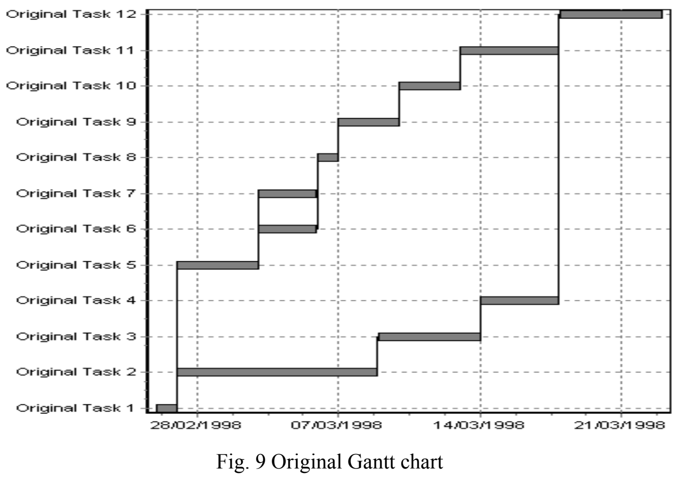

Fig. 9 Original Gantt chart

The proposed parallel FCM tool was applied to simulate a real project related to schedule of time and resources in the web design, which is described in [8]. The guidelines for project and a Gantt chart, which is shown in Fig. 9, are borrowed from [8]. The Gantt chart describes an autonomous part of whole project, and is used to perform experiments with the parallel FCM tool.

The descriptions of all tasks along with their duration and start time are specified in Table I. Comparing to the origin schedule, some dates were changed by considering the weekends as work days, whereas they were treated originally as holidays.

TABLE I
LIST OF TASKS

Task number	Description	Duration (days)	Start time	Type[a]
1	Write design brief, communicate to staff	1	Feb 26	B
2	Writing: first draft complete	10	Feb 27	B
3	Writing: editorial review of draft	5	Mar 9	B
4	Writing: final draft complete	4	Mar 14	A
5	Define navigation and site structure	4	Feb 27	C
6	Complete sketches for graphic elements and placement of media	3	Mar 3	C
7	Complete sketches for interface design	3	Mar 3	B
8	Review sketches and site structure	1	Mar 6	A
9	Create low-fidelity prototype of interaction and navigation	3	Mar 7	A
10	Test prototype with users	3	Mar 10	B
11	Visual and media elements complete (first iteration)	5	Mar 13	B
12	Create high-fidelity prototype including content	5	Mar 18	C

[a] A – unpartitionable task
B – partitionable task requiring communication
C – task with complex interrelationships

The type of tasks was selected based on characteristics of particular tasks, which were derived from their description. Each task is also described by a number that expresses its complexity, and which represents the amount of work that has to be done in order to complete it. It corresponds to the number of days depicted on the Gantt chart. Complexity of each task (amount of work to complete the task) is equal to half of its duration expressed in days. This relationship was reported in [3].

The above project was simulated using the parallel FCMs architecture. Each simulation considered different set of initial conditions, which correspond to different assignment of workers to particular tasks.

First simulation was performed by assigning workers without considering the task type. In this case the available manpower was evenly assigned to the tasks. For example, if upon completion of a given task the workers would be divided into two new tasks, then each new task would be assigned half of the workers. To enable modelling projects with an arbitrary number of workers, the total number of workers was normalized to 1. Considering these assumptions, the initial values of people base nodes, which represent the amount of workers assigned to each task, for the first experiment are shown in Table II.

TABLE II
INITIAL VALUE OF PEOPLE BASE NODE FOR EACH TASK IN SIMULATION 1

Task	1	2	3	4	5	6	7	8	9	10	11	12
Initial value	1	0.5	0.5	0.5	0.5	0.25	0.25	0.5	0.5	0.5	0.5	1

Result of this simulation, i.e. Gantt chart, is presented in Fig. 10. The grey chart shows the original schedule, while the black one shows the simulated schedule. The ending points of grey (Original Task 12) and black chart (Task 12) correspond to original and simulated project end date, respectively.

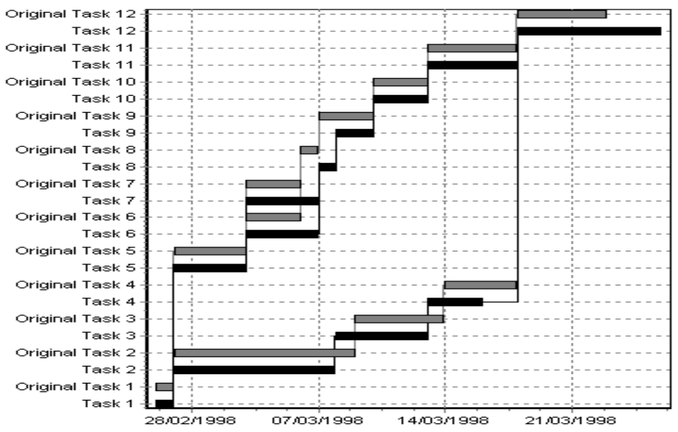

Fig. 10 Gantt chart obtained from simulation 1

As can be seen from this chart, the simulated schedule is worse than planned by three days, which is caused by the simplistic assignment of workers to tasks. Several others simulations were performed with different initial conditions. These conditions were chosen on the basis of the following heuristics:

A. For A-type tasks the initial value of people base has no significant impact on progress, and thus it can be set at a low level.
B. For B-type tasks increasing in number of people is justified, so the amount of assigned workers should be maximized.
C. For C-type tasks there exists a limitation above which increasing people base value has negative impact on task duration, so amount of assigned workers should be high, but not exceeding the limit.

By applying above heuristics, the simulated schedule was significantly improved, when compared with the results of the first simulation. The parameters and Gantt chart that describes the simulations with the shortest resulting schedule are presented in Table III and Fig. 11, respectively. The improved assignment of workers resulted in the simulated schedule that is shorter than the original schedule.

TABLE III
INITIAL VALUE OF PEOPLE BASE NODE FOR EACH TASK IN SIMULATION 2

Task	1	2	3	4	5	6	7	8	9	10	11	12
Initial value	1	0.4	0.5	0.4	0.5	0.3	0.3	0.4	0.4	0.5	0.5	0.5

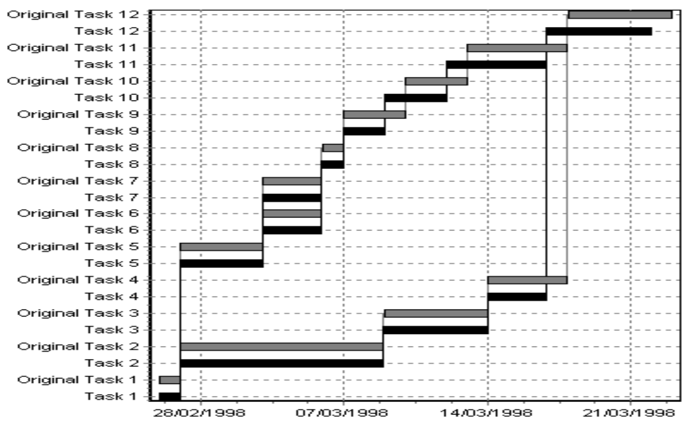

Fig. 11 Gantt chart obtained from simulation 2

Comparison the results of the first and the final simulation shows that it is important to identify type of each task before starting software project. It is also essential to recognize that adding more people can boost only some of the tasks. We note that based on these observations, it is possible to perform optimization of worker assignment, which will results in decreasing the overall length of the project. On the other hand, neglecting these observations may lead to mismanagement of human resources. This, in consequence, may lead to project completion delays. To show the impact of incorrect worker assignment decisions another experiment was performed. The Table IV shows initial conditions that are chosen against the presented above heuristics. Corresponding simulation, which is presented in Fig. 12, shows the resulting one week delay of the project.

TABLE IV

INITIAL VALUE OF PEOPLE BASE NODE FOR EACH TASK IN SIMULATION 3

Task	1	2	3	4	5	6	7	8	9	10	11	12
Initial value	1	0.2	0.2	0.2	0.8	0.4	0.4	0.8	0.8	0.8	0.8	1

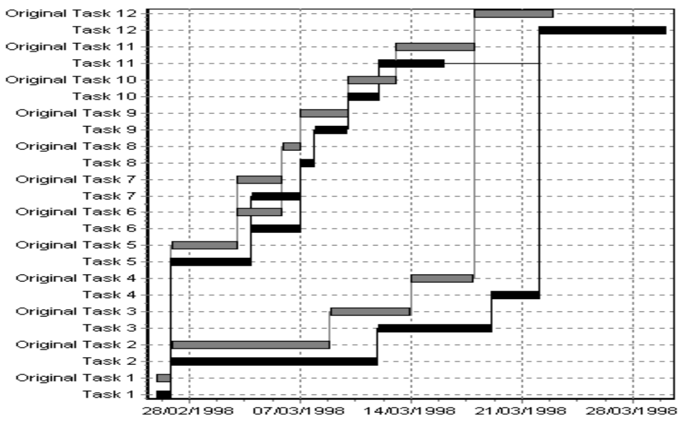

Fig. 12 Gantt chart obtained from simulation 3

V. CONCLUSIONS AND FUTURE WORK

The paper proposes a novel parallel Fuzzy Cognitive Maps (FCM) architecture. The architecture is suitable to simulate dynamic systems, which include a number of sub-systems that run in parallel, and are represented by individual FCM models. The proposed architecture allows analysis of the behavior of both entire system and each FCM individually. The simulation of the proposed parallel FCM model allows for exploratory analysis of the simulated system.

The paper also presents successful application of the newly proposed model to simulate scheduling of software projects. Three individual FCMs that describe different software task types were developed. The software project is represented using a Gantt chart that shows relationship between all atomic tasks that constitute the project. The experiments that aimed to simulate the anticipated schedule for different assignment of workers for the atomic tasks showed that parallel FCM is a helpful tool to simulate such complex dynamic systems. Analysis of different simulated schedule scenarios provides valuable support that helps to determine and minimize length of the schedule based on the worker assignment.

REFERENCES

[1] G.D. Boetticher, "Using Machine Learning to Predict Project Effort: Empirical Case Studies in Data-Starved Domains", *Model Based Requirements Workshop*, San Diego, CA, pp. 17 – 24, 2001

[2] G.D. Boetticher, “When Will It Be Done? Machine Learners Answer the 300-billion-dollar Question”, *IEEE Intelligent Systems*, pp.2-4, June 2003

[3] F.P. Brooks, JR., “The Mythical Man-Month”, *Addison-Wesley,* 2001

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

[5] S. Flower, “Software Failure, Management Failure”, *Wiley & Sons,* 1996

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

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

[8] IBM, Schedule of Time and Resources for Web Design Task, available at http://www-3.ibm.com/ibm/easy/eou_ext.nsf/Publish/599

[9] IEEE Standard for Software Project Management Plans, ANSI/IEEE Std 1058.1-1987

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

[11] B. Kosko, “Neural Networks and Fuzzy Systems”, *Prentice-Hall*, 1992.

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

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

[14] W. Stach, and L. Kurgan, “Modeling Software Development Project using Fuzzy Cognitive Maps” *Proc. of the 4th ASERC Workshop on Quantitative and Soft Software Engineering (QSSE'04)*, pp.55-60, Banff, AB, 2004

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

[16] The Standish Group, *CHAOS Chronicles*, Standish Group Internal Report, 1995

[17] C. Stylios, and P. Groumpos, “The Challenge of Modeling Supervisory Systems using Fuzzy Cognitive Maps”, *Journal of Intelligent Manufacturing*, vol. 9, no. 4, pp.339-345, 1998

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