CellS: A cell-inspired efficient software framework for Intelligent System

Mu-Che Wu and Ching-Han Chen
National Central University, Taoyuan, Taiwan

Abstract-- **Imagine next generation system can talk with people, move around, aware of and respond to changes in the environment, and learn from data. This system needs a lot of sensors and effectors in its hardware. In its software, it must have the capability to handle sensor data autonomously and to generate appropriate responses. This kind of system exactly is BDI (Belief, Desire, and Intention) system. However, traditional BDI system is too difficult to develop a software system, and it needs more cost in maintaining and developing software. To reduce the cost and increase efficiency, we propose a cell-inspired efficient software framework called the CellS. The framework implementation adopted the concept of traditional BDI system refactored by the new cell layer. The cell design can illustrate the BDI concept specifically. Software architectures continue evolvement, from structured programming to object-oriented programming, from single processor programming to multi-processor programming, from passive calls to active execution. Our framework adopted dynamic parallel mechanism, autonomic BDI and Java language. It suits to all java development environments, and developer just need to learn basic BDI concept. The CellS has the following advantages: 1) high performance; and 2) high scalability. An empirical experimental evaluation demonstrates that the CellS work well in performance.**

I. INTRODUCTION

Software is constantly evolving, in order to meet the expectations of users, from web to Android, from PC to IOT. The next stage of the software is bound to need to sense the needs of users, and can provide their services autonomously. In order to provide such services, the capability of perceived environment and autonomy should be the most basic needs of the next stage of the software architecture. The BDI software architecture coincides on this demand [1]. BDI is currently one of the most promising approaches for the development of complex systems [2]. It is also a kind of solution for complex interactive reasoning problems [3]-[5]. There are many tradition BDI-based agent platforms, such as JACK, Jason, JAM and JadeX [6]. Various technologies, such as network intrusion detection [2], teleo-reactive systems [7], Ubiquitous computing [8], [9], use BDI system, too.

In this article, we provide a cell-inspired efficient software framework to reduce the cost of software developing and maintaining. This framework based on the concepts of BDI and cell theory, using pure Java without special script. Therefore, it is applicable to all Java environment development, even if it is in Android environment is workable too. In the software design, software engineers get used to designing in a sequence flow. However, the BDI system to perform by dynamically selected a plan, and its concept is similar to jumping thinking. How to design a software architecture that combines these two thinking is a challenge to design the software framework. We inspired by cell theory, and the results of our implementation prove that such an approach is indeed feasible. The inspired descriptions are as follows:

1. Any system constructed by the CellS is composed of one or more cells.
2. The cell is the basic unit of structure and organization in the CellS.
3. All cells arise from pre-existing cells.

II. THE ARCHITECTURE OF CELLS

We use MIAT methodology and Grafcet to establish the architecture of the CellS framework, shown in Fig. 1. This framework divided into two main parallel sections. The first part is using sense cells to sense and collect data from the external environment. The second part is triggering brain cells and selecting plans by external environment data. All selected plan will executed at the same time. Each plan is an implementation to complete the purpose of its brain cell. Different plans in the same brain cell represent different ways of implementing brain cells in different situations. A plan will use motor cells to complete its purpose of brain cells. Each motor cell will affect or change the external environment. When the CellS framework does not terminated (i.e. it is alive), sense cells will proactively continue to detect the environment after all selected plans have completed their work. Completing two parallel sections, called a life cycle of the CellS.

A more detailed description of the CellS architecture is as follows. The total number of sense cells is n. The sub-Grafcet Si represents the i^{th} sense cell executes its perception task, where i is a nature number between 1 and n. The total number of brain cells is m. The sub-Grafcet Bj represents the j^{th} brain cell selects a plan and execute it, where j is a nature number between 1 and m. The condition C0 means the condition that starting the CellS framework. The condition C1 means the condition that all sense cells finish their tasks. The condition C10 means the condition that starting all sense cells. The condition C11 means the condition that transforming all information into beliefs. The condition C2 means the condition that all brain cells finish their tasks. The condition C20 means the condition that starting all brain cells. The condition C2j means the condition that the j^{th} brain cell has a desire which need to finish. The condition C2(m+1) means the condition that all brain cells finish their tasks.

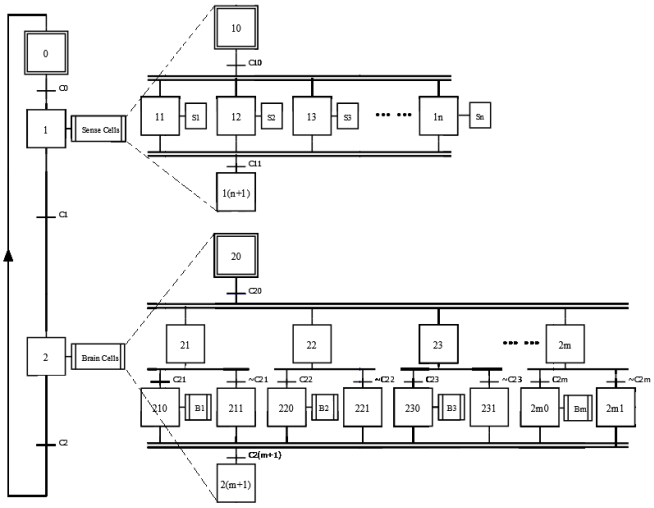

Fig.1. The architecture of CellS.

III. Application and Experiments

In this section, we build a simple dialog system on Android to show how to use the CellS framework to create an application. In the other hand, we proceed to demonstrate the performance of our framework. To do this, we execute applications on a Qualcomm Snapdragon 810 and a Qualcomm Snapdragon MSM8916 410, and compare their actual execution time of the application's associated sequential programming and the CellS framework. In the experiment, each task set includes 100 tasks and those tasks arrived to dialog system at same time. Table I shows the original experiment result.

TABLE I

Result of Experiment

Task Set	Qualcomm Snapdragon MSM8916 410		Qualcomm Snapdragon 810	
	Sequential Programming	CellS Framework	Sequential Programming	CellS Framework
1	86211 ms	62766 ms	69473 ms	41146 ms
2	83934 ms	53326 ms	95372 ms	55205 ms
3	99214 ms	66898 ms	91583 ms	55833 ms
4	93513 ms	85395 ms	78924 ms	58165 ms
5	79042 ms	68051 ms	59737 ms	46530 ms
6	79040 ms	64000 ms	64684 ms	51213 ms
7	99820 ms	62245 ms	96072 ms	50634 ms
8	95197 ms	57784 ms	77652 ms	46955 ms
9	83362 ms	57132 ms	83296 ms	48736 ms
10	83291 ms	57358 ms	103904 ms	53446 ms
Average	88262.4 ms	63495.5 ms	82069.7 ms	50786.3 ms

From our experimentation, we can conclude that in multiple cores platform the CellS framework speed up the dialog system and increase its throughput of task sets. The performance of our framework is better if platform has more number of cores. Regardless of what application in the multiple cores platform, our technique is always able to speed up the application and increase its throughput.

IV. Conclusion

In recent years, IOT (Internet of Things), Robot and AI research are widely by attention of people, and the BDI concept applies to these areas of research. Therefore, a software framework with usability is particularly important. In this study, we developed a framework called the CellS with reference to the traditional BDI and inspired by the cell theory. Due to the CellS design, software engineers can easily focus on business logic development and cross the threshold of BDI. It is a fast and intuitive development framework. The software framework with reference to the traditional BDI system has a well dynamic and extensibility, due to combined with the parallel computing and the heuristics of the cell theory, we build the dynamically parallel mechanism in it. The results of the experiment can prove that this framework has the advantages of high performance. Our future research will include how to use CellS in IOT, Robot, and AI, thereby increasing their practical benefits.

Reference

[1] A. S. Rao and M. Georgeff. BDI Agents: from theory to practice. In Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95), pages 312–319, San Francisco, CA, June 1995.

[2] A. Leite and R. Girardi, "A hybrid and learning agent architecture for network intrusion detection", Journal of Systems and Software, vol. 130, pp. 59-80, 2017.

[3] M. Al-Jarrah and H. Almasaeid, "AN ARCHITECTURE OF INTELLIGENT MULTI-AGENT SYSTEM: A CASE STUDY IN SOCCER GAME SIMULATION", International Journal of Modelling and Simulation, vol. 30, no. 1, 2010.

[4] J. Kuo and F. Huang, "A Hybrid Approach for Multi-Agent Learning Systems", Intelligent Automation & Soft Computing, vol. 17, no. 3, pp. 385-399, 2011.

[5] J. Kuo, F. Huang, S. Ma and Y. Fanjiang, "Applying hybrid learning approach to RoboCup's strategy", Journal of Systems and Software, vol. 86, no. 7, pp. 1933-1944, 2013.

[6] Rafael Bordini, Lars Braubach, Mehdi Dastani, Amal El Fallah Seghrouchni, Jorge Gomez-Sanz, Joao Leite, Gregory O'Hare, Alexander Pokahr, and Alessandro Ricci. A survey of programming languages and platforms for multi-agent systems. In Informatica 30, pages 33–44, 2006.

[7] P. Sánchez, B. Álvarez, J. Morales, D. Alonso and A. Iborra, "An approach to modeling and developing teleo-reactive systems considering timing constraints", Journal of Systems and Software, vol. 117, pp. 317-333, 2016.

[8] D. Zhang, J. Wan, X. Liang, X. Guan, Q. Liu and G. Ji, "A taxonomy of agent technologies for ubiquitous computing environments," KSII Trans. on Internet and Info. Syst., vol. 6, no. 2, pp. 547-565, Feb. 2012.

[9] K. Gunasekera, A. Zaslavsky, S. Krishnaswamy and S. Loke, "Building ubiquitous computing applications using the VERSAG adaptive agent framework", Journal of Systems and Software, vol. 86, no. 2, pp. 501-519, 2013.