Proceedings of the International Workshop on Intelligent Information Systems
IIS'2011, September 13-14, 2011, Chisinau, Republic of Moldova

Concurrency Anatomy of Intelligent Information System Based on Agents

Victor Besliu, Dumitru Ciorba,
Sergiu Ciumac, Vadim Andronachi

Abstract: Currently, modern applications cannot be imagined without intelligent entities, which imply high performance requirements. This can be achieved using concurrent techniques. Even though, all parallel activities present within a system, increase its behavioral complexity. That's why there is a strong need in adequate architecting that can significantly decrease implementation errors. Additionally, in this paper we would like to present an increasingly popular concurrency methodology that is used alongside with AI specific solutions: agent programming model.

Keywords: intelligent information system, concurrency-intensive architecture, actor model, agent programming model.

1 Intelligent Information Systems (IIS)

Main driving forces of a modern society evolution to one based upon knowledge, are the scientific-technical progress and information viewed as a production factor. J. A. O'Brien [1] presents today's society as being in a transition period covered by three technology waves, the latest stage being the *global information society*. In the perspective of the classic relationship "data-information-knowledge" authors of [1] suggest that the next stage will be the stage of knowledge. This stage will be characterized by intelligent information systems, focused on the information exploration, to achieve a desirable level of intelligence by some system entity.

Benefits of modern multiprocessor systems can be exploited to meet performance requirements only by naturally representation of concurrent activities. An adequate example may be agent-based systems, which are inherently concurrent. It should be noted then, that the concurrency is considered as a critical property of such systems, and that it must be considered in the early development stages – architecting stages.

2 Concurrency-intensive architecture

According to IBM *Rational Unified Process*™ architecture is defined during the inception and elaboration phases [2]. This popular software development process is architecture centric. It means that the system's architecture is a primary artifact for system's development [3]. Thus the importance of an accurate architecting must be sustained by a distinct process. In this context a generic framework is proposed below. It will permit us analyzing concurrency-intensive architecture [4] from the perspective of evolution (see Fig. 1). Each phase defines an architecture model, which may also include a number of views, where each view is related to a particular domain of the phase.

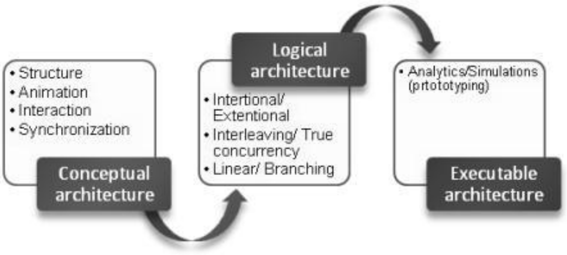

Figure 1. Concurrency-intensive software architecture evolution

Conceptual architecture defines entities, their relationships, over which are defined rules to synchronize, select and accept operations, and these rules define control constrains used at the next *Logical architecture* phase. *Logical architecture* conforms to the principles and rules of the conceptual architecture. This phase involves a variety of structures, which have the nature of mathematical formalisms, determining the *logics of specification,* which helps describing and reasoning about behavior of concurrent architecture. *Executable architecture* is a result "product" of the last phase. This term may express the description of system's architecture in a formal notation, the semantic of which is being determined by the logic phase, or may signify a partial implementation of the system – a prototype.

3 Anatomy of IIS in the scope of concurrency

Anatomy of IIS can be analyzed in the perspective of concurrency conceptual views of described architecting framework. The *Structure view* can show configurations in terms of components, which are units of runtime computation or data-storage, and connectors, which in their turn are the interaction mechanisms between components [5].

The next three views have been inspired by a survey of concurrency issues presented in [6]. The survey is organized from the taxonomy of features of concurrent object-oriented languages. However the generalization of the described models, allows us to use them in our architecting process as views. *Animation view* shows the relationship between objects and active entities (process/thread/task). The treatment of threads and objects as independent or dependent concepts, defines two alternatives of activity organization: unrelated and related models. The *interaction view* depicts interactions between objects initiated by the client's invocation, which may be either synchronous or asynchronous. Concepts, related to *Synchronization view*, specify concurrent invocation management.

4 Actor model as concurrent model of agent-based IIS

Popular agent-based systems require a platform, which can define data pipelines or networks. Data pipeline is a formal approach, according to which primary goal is achieved by dividing processing specific tasks between agents (components, presented in *structure view*) [3].

The Actor model can provide an adequate interaction infrastructure which meets all requirements: an actor can implement an independent orthogonal functionality (related model, presented in *animation view*); communication model is asynchronous (presented in *interaction view*) and is based on dataflow instead of control flow (communication is done by transferring messages), etc. [7]. As the result, the agent programming model has several advantages: scalability; extensibility; implicit parallelism; asynchronous model, easy isolation of access to shared resources; etc.

Now, there are many modern actor/agent based libraries, frameworks and languages, which permit agent-style programming. Among these languages are: Erlang (Ericsson), Scala (Martin Odersky, EPFL), F# (.Net/Microsoft), Haskell, etc.

5 Conclusion

As concurrent systems become more complex there is a strong need in defining new methodologies which will solve specific data flow problems. Concurrent agent approach gives the researchers the advantage of focusing on the problem that needs to be solved rather than how these specific issues are tackled. Applications which exhibit a complex internal/external data flow can be successfully parallelized using the approaches described in this paper. As any methodology it doesn’t represent a panacea to all AI specific problems, as control flow solutions are still subject of an area of classical concurrency.

References

[1] M. Vlada and Al. Tugui. *Information Society Technologies - The four waves of information technologies*. The 1st International Conference on Virtual Learning (ICVL). 2006. http://fmi.unibuc.ro/cniv/2006/disc/icvl/documente/pdf/met/1_vlada.pdf.

[2] IBM Rational Software. *Rational Unified Process. Best Practices for Software Development Teams*. IBM developerWorks®. [Online] July 23, 2005. [Cited: May 11, 2011.] http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf.

[3] H. Yim, et al. *Architecture-Centric Object-Oriented Design Method for Multi-Agent Systems*. Fourth International Conference on Multi-Agent Systems (ICMAS'00). Los Alamitos, CA, USA : IEEE Computer Society, 2000. ISBN: 0-7695-0625-9.

[4] D. Ciorba and V. Besliu. *Architecting software concurrency*. Computer Science Journal of Moldova. 2011, Vol. 19, 1 (55).

[5] P. Clements. *Documenting software architectures: views and beyond*. s.l. : Addison-Wesley, 2003. ISBN 0-201-70372-6.

[6] D.G. Kafura and G. Lavender. *Concurrent Object-Oriented Languages and the Inheritance*. [ed.] T.L. Cassavant. Parallel Computers: Theory and Practice. s.l. : IEEE Press, 1994. pg. 165-198.

[7] G. Agha, and P. Thati. *An Algebraic Theory of Actors and Its Application to a Simple Object-Based Language*. Formal Methods and Declarative Languages Laboratory. [Online] 2004. [Cited: 26 November 2010.] http://formal.cs.uiuc.edu/papers/ATactors_festschrift.pdf. LNCS 2635.

Victor Beșliu[1], Dumitru Ciorbă[2], Sergiu Ciumac[3], Vadim Andronachi[4]

Technical University of Moldova
[1]E-mail: besliu@mail.utm.md
[2]E-mail: dumitru.ciorba@ati.utm.md
[3]E-mail: ciumac.sergiu@gmail.com;
[4]E-mail: andronachi.vadim@gmail.com