15th Triennial World Congress, Barcelona, Spain

AN ARCHITECTURE FOR A MULTI-AGENT SYSTEM TEST-BED

L. Motus*, M. Meriste, T. Kelder**, J.Helekivi****

**Tallinn Technical University, Estonia*
leo.motus@dcc.ttu.ee
***University of Tartu, Estonia*
merik.meriste@ut.ee

Abstract: Architecture of a test-bed is suggested that has capabilities for the development of agents and multi-agent systems, and also serves for semi-automatic testing, assessment and verification of selected properties of those agents and systems. The test-bed is essentially based on models of interactive computations. New features in the test-bed are caused by an attempt explicitly to describe and formally analyse timing characteristics of agents and their interactions. Time properties of agents and their interactions have top importance in guaranteeing proper functioning of many monitoring and control applications. *Copyright © 2002 IFAC*

Keywords: Agents, computer control, multi-stream interaction machine, timing analysis, test-bed

1. INTRODUCTION

The co-operation and communication capability of heterogeneous computing systems is rapidly increasing due to progress in elaboration and approval of standards by W3C and OMG (see www.w3.org and www.omg.org respectively). The achieved progress facilitates a wider application of loosely coupled software systems, like agent-based systems, in the industrial environment. Technologies of the last decade have focused on the interaction between the agents, and of agents and their environment.

For instance, in addition to traditional mobile robots and several logistic applications, agent-based technology is increasingly considered for distributed computer control systems, remote condition-monitoring and diagnosis systems, and other (possibly collaborative) decision-making applications. This is partially caused by the capability of interacting agents to generate more complex behaviour than one would expect from the straightforward structure of agents.

The application domain of agent-based paradigm is extending. This process inevitably causes the emergence of "latent" requirements to agents and, especially to multi-agent systems – such as history dependence and explicit time-sensitivity of their behaviour. In many cases empirical demonstration of expected behaviour may not suffice. Instead, formal verification of behaviour is desired, this in turn assumes the use of models of interactive computing (see, for example Wegner and Goldin 1999, Motus 1995).

Also, in the increasing number of applications it becomes essential that several time constraints be verifiably satisfied by each agent, and by groups of co-operating agents (Motus and Meriste 2001). This introduces new aspects into a discussion regarding models of interactive computations, time models and their combined use in verifying time-behaviour in software-intensive systems (e.g. Motus and Naks, 2001).

Relatively close to interactive computing is the phenomenon of emergent behaviour (Simon, 1996) in multi-agent systems – e.g. real-time systems, or human organisations. In such systems agent's learning implies adaptation and adaptation in turn implies interaction. Adaptation (pro learning) is an

example of history-dependent behaviour that cannot be modelled within an algorithmic model (i.e. without history), neither in an order-of-events setting (Goldin and Keil, (2001)). The premises and methods, necessary to satisfy the emerging "latent" requirements have never been sufficiently investigated. In some cases qualitative recommendations can be given, however, quantitative recommendations, as well as the underlying theoretical study are not yet available. Time-constraint negotiations, dynamics of learning and adaptation, and their influence upon the goal function are just some examples of insufficiently studied problems. The essence of negotiation process (see, for example, Rosenschein and Zlotkin 1998, Bailin and Truzskowski 2001), learning (e.g. Stone 2000) and adaptation (e.g. Tsypkin 1971) has been studied relatively thoroughly – still, the influence of time constraints on negotiating, learning and adaptation capability is not really studied.

Let us imagine negotiations concerning the potential co-operation, or competition of two agents, under incomplete information about the goals and capabilities of respective partners. It seems pretty obvious that the one with better negotiation strategy, or faster learning curve and/or better adaptation dynamics has serious advantages for fixing winning conditions during a time constraint negotiation process.

Another group of emerging requirements is bound to the model of computations on which the verification methods are based. In addition to seminal papers by Wegner (1997), Wegner and Goldin (1999) and Goldin and Keil (2001), several interesting results (at least implicitly) related to interaction-based computing are scattered around without proper comparison of properties and cross-reference to each other. A variety of models of interactive computations have emerged from three independent application domains. For instance, from distributed artificial intelligence (kenetics -- (Ferber (1999) and *MadKit*) and rational agents -- Wooldridge (2000)), from computer science and software engineering (formal program verification -- Milner (1999) and object-oriented programs -- Wegner (1997)), and from modelling real-time software (timing analysis -- Motus (1995), Caspi and Halbwachs (1986)).

The models of interactive computations emphasise and support verification of systems where a primitive component for building a system is not an algorithm, but rather a set of interacting, repeatedly activated, terminating algorithms; or alternatively a non-terminating computing process. In many cases the study of infinite, and dynamic input/output sequences can reveal some properties of the respective components – i.e. regarding non-terminating behaviour of computing processes, as well as the behaviour of countable many times repeatedly activated, terminating, interacting computing processes.

From a more practical point of view, many interesting methods and supporting tools for developing agents and multi-agent systems have been developed. Majority of those methods and tools focus on studying specific agent-related properties, and on developing agent-based applications (see, for instance, *MadKit* and *Jade*). Overwhelmingly the study of properties is based on empirical comparison of certain features of agent's behaviour to those observed in biological individuals and/or social organisations formed by biological individuals.

Agent applications are prevailingly organised by attempting to find and maintain a suitable order of interactions and decisions. However, in increasing number of applications mere qualitative ordering is not sufficient -- e.g. time-critical applications. Quite often ordering of events assumes complete knowledge of causal reasons. In some cases this knowledge may not be available, or may not be usable because of insufficient computing power -- in that case one can always approximate causal reasons by time-constraints.

In this paper the authors suggest an approach to the development of agents, agent systems and the supporting development and analysis tools that differs from the conventional one in two aspects. Firstly, this paper suggests that time sensitivity of various parts of agents, and agents' interactions, is to be explicitly described and studied by using formal models of interactive computations.

Secondly, the authors suggest that sufficiently good theoretical and practical basis exists to start building a test-bed that would foster practical use of formal analysis methods and thus assist in merging recent development trends in computer science, software engineering, artificial intelligence, computer control, and other domains. Such a test-bed would serve as a discovery system (Evans, 2001) for agent-related knowledge, in addition to being just a development environment for agents and agent systems. For instance, the test-bed would serve as an experimental basis for elaboration of theories and methods for quantitative assessment of selected properties, such as efficacy of the negotiation process, and evolution of communication capabilities of the structure that is emerging during the negotiation process and/or interaction of agents. A special tool in a test-bed is to take care of verifying the consistency of time-behaviour in multi-agent systems.

2. ARCHITECTURE OF THE TEST-BED

The test-bed itself is a multi-agent system, and therefore comprises loosely coupled components (agents) together with a framework of basic services and only partially defined interaction structure between those components. The test-bed is modelled in UML and development is based on unified

software process, standards and recommendations provided by FIPA are carefully followed. However, the authors are aware, that the development of such a test-bed is kind of a non-terminating process. Many methods and tools required for the test-bed are not readily available, and the test-bed itself serves as an inspiring generator of facts and requirements for fostering the research into methods of operation assessment of multi-agent systems.

2.1 Pragmatic point of view

The test-bed comprises a shell that provides basic common services, agents registered with the shell, explicitly registered coalitions of agents, and various tools used to develop agents and to investigate their behaviour. It is essential that the agents can reside in geographically distributed locations also the tools may be distributed, if necessary.

The shell of the test-bed provides an extendable framework of basic common services. For instance, directories for agents registered with the shell, support for creation of repositories for common knowledge to be used by all agents, and/or by coalitions of agents (see Fig.1). Separate set of services is provided for the monitoring, assessment and analysing tools, such as knowledge- and databases created or used by the tools, the communication services for interaction of tools and agents. The provided services can be invoked and accessed automatically by the agents and/or tools according to assigned access rights, or in some cases by the user via the main user interface of the test-bed. The access rights are requested and granted via the main user interface. From the main user interface one can also navigate to specific tools, to repositories, knowledge- and databases that in many cases may have their own user interfaces.

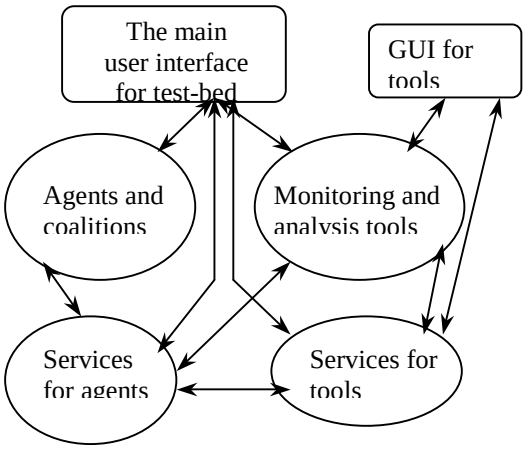

Figure 1. High-level structure of the test-bed

2.2 Idealistic point of view

The test-bed is a multi-agent system comprising three, or more clusters of agents (or agent-like components). Agents belonging to the same cluster have, as a rule, full-scale in-cluster communication capability. However, each agent can freely select the level of its in-cluster co-operation with the other agents. There are no general restrictions on shared information and depth of co-operation between agents in a cluster, since the cluster members are closely collaborating and their goal functions are consistent with each other by definition. A cluster may be partitioned into sub-clusters (coalitions), if the need for general restrictions appears. Usually this means that agents from different sub-clusters compete with each other, and the restrictions cater for safety, security, and other application-specific rules of behaviour.

Each cluster has a different role each (sub)-cluster unites agents that have consistent roles. For instance, one cluster comprises (or models) the real-world entities, or strange agents that cannot be optionally modified by the designer of the other (sub)-clusters. In some cases this cluster may include a physical model, or actual physical agents (e.g. mobile robots).

The second cluster models (or represents) agents that monitor or control agents from the first cluster, or interact with each other in order to satisfy given goal functions. Competing coalitions (sub-clusters) of agents are possible within this cluster. Conventional multi-agent development systems usually focus on problems that are strictly intrinsic for this cluster of agents.

The third cluster comprises development, measurement, estimation and reasoning tools used for assessing the properties resulting from the evolution processes in the two previous clusters, plus the tools supporting description of agents forming the two previous clusters.

2.3 About the tools

The tools will be added to the test-bed gradually, as the test-bed and the tools evolve. The first tool is for agent development – a standard tool will be used, (e.g. *Jade*) with some modifications in order to enable the use and comparison of various models of interactive computation (e.g. sequential and multi-stream interaction machines and Q-model (see Motus and Meriste 2001)).

The other tools (e.g. for timing analysis; see Motus (1995)) are to be adapted for the agent systems, or are to be developed from the scratch, as the theoretical basis for the analysis becomes ready (e.g. a tool for monitoring the evolution of integral behaviour of a multi-agent system). The timing analysis tool has been tested for many years (see, for instance, Naks and Motus 2001) and is, in principle, suitable for analysing timing in a multi-stream interaction machine. However, timing in agent systems is, in

many cases, subtler and more complicated issue as compared to timing in a conventional real-time system.

In a conventional real-time system one may need to consider different time counting systems and time concepts in different subsystems. In time-sensitive agent-based systems each agent may need three separate time-counting systems (for communicator, manager and functional body, see section 3 of this paper). Within each time-counting system strictly increasing thermodynamic, fully reversible, and relative time concepts should be present to guarantee full timing analysis capability (Motus and Rodd, 1994).

Potentially required modification of the timing tool may invoke additional research into timing analysis. Timing analysis is done off-line (e.g. pre-tun-time scheduling) as a rule, based on required, specified and actually measured time properties (and theoretical models) of specification and design of the multi-agent system. The interaction structure of the agents and traffic estimates reached up to the moment of timing analysis as the result of on-going adaptation and learning should also be considered, especially if the interaction structure differs from that used when the previous timing analysis was performed. In highly dependable applications one may also need run-time monitoring and/or diagnosis of timing correctness.

Similar approach can be used for assessing the efficacy of negotiating, learning and adaptation algorithms. The case of monitoring and analysing a test run of the designed system (i.e. the actual co-operation of agents, environment, and possibly strange agents) in order to assess the satisfaction of the goal function, is more sophisticated and definitely needs some online measurements. This in turn assumes the existence of measuring subsystem (agent) and a mechanism for making the measurements accessible to tools without violating the actual behaviour of the monitored system.

The basic foreseeable problem invoked by on-line measurements is the potential violation of time constraints imposed upon the behaviour of an agent system due to added actions related to on-line measurements and monitoring. The problem is not serious if the on-line assessment of multi-agent system's functioning is done in a simulation mode -- this would allow to filter out the activities related to monitoring and measuring.

3. ARCHITECTURE OF AN AGENT

An agent has to meet several expectations, some of which can be considered as requirements that cannot be avoided. Some instances of unavoidable requirements are -- a registered agent has to comply with FIPA standards (see *FIPA*), and with the other specific requirements of the test-bed shell. However, one of the interesting research areas is interaction with strange agents – this would be possible only if the test-bed allows exceptions. Agents that do not meet FIPA standards, or some other common agreements within the test-bed, should be allowed to register with the test-bed as an exception under special security measures.

The other, less generally applied, expectations regarding agent's architecture stem from the subjective research goals of the authors. Those goals include study and comparison of the properties of alternative implementations of a multi-stream interaction machine, description of time sensitivity of various parts of an agent, and reasoning about the influence of time properties of member agents upon the operating efficacy of agents' coalitions.

The above considerations have resulted in a logical structure of an agent as comprising three closely interacting but relatively autonomous and independently substitutable parts (components):

- *Communicator*, that interacts with the other agents via the respective functional part in the test-bed shell; performs the first-level interpretation of the messages, and is responsible for carrying out the negotiations from the first contact to the termination, according to the approved protocol;
- *Manager*, that performs more advanced interpretation of the message contents, updates the estimates of beliefs (or knowledge) of the agent; modifies the respective knowledge base about the agent's actual and potential partners, decides about the proper response to messages, adapts the functions of the agent's main body according to the results of the negotiations, and adjusts the goal function of the agent if necessary;
- *Functional body* of an agent is responsible for proper actions that lead to satisfaction of agent's goal function, operation of the *functional body* can be reorganised and/or adapted by the *Manager*. The adaptation is based on the new knowledge extracted by the *Manager* from the messages of other agents, the environment, and by recent values of the goal function.

Each of those components may have several ready-made substitutes (using alternative methods) in the test-bed repositories. The researcher (or user) that works with the test-bed can select a substitute for a component in order to compare the influence of different methods. A vaguely similar approach has been taken in Fricke et al (2001). Such a partition will assist in theoretical and experimental study of the influence of time-sensitivity and adaptability of an agent upon overall functioning of multi-agent systems, since each separate time sensitive and adaptive component of an agent can be modified independently of the others.

From the point of view of agents' timing analysis this also illustrates the increase in complexity as compared with conventional real-time systems – in many cases one has to deal with three explicitly separate time models within one agent. Communicator, manager and the functional body of an agent may operate, in a general case, within different time-scales based on different time-counting systems.

A test-bed has dual role – it supports the development of multi-agent systems, and in the same environment the developer can test, formally analyse, and assess different aspects of the system's operation by using a suite of dedicated tools. This becomes possible only if those tools have explicit access to intrinsic variables of the multi-agent system – e.g. state variable values, parts of messages, updated beliefs, and other information that influences the decisions. Such information is usually not available for the outside world of an agent (or a group of agents). The extraction of intrinsic information and its transportation to external user during normal operation may cause serious technical difficulties, e.g. security problems and also coherence of obtained information is often questionable. In the test-bed this become possible when the monitored multi-agent system operates in a simulated time.

In addition to directly addressed inter-agent messages an agent consumes and produces information for intrinsic use, such as messages for internal information exchange between communicator, manager, and functional body of the agent. Quite often an agent needs a regularly updated in-agent knowledge base to support intrinsic decisions required for controlling and adapting its operation.

Each coalition of agents, and the whole multi-agent system usually creates common knowledge base that can be freely accessed by coalition members and/or by system members. Typically this base contains variables' values and events occurring in the environment, rules of conduct approved by the environment and other more specific knowledge that may be different for each coalition of agents.

It is not quite clear what is the most suitable way for the tools to access the information that actually determines the behaviour of multi-agents systems. Slightly adapted blackboard technique has been selected as a starting point for the test-bed. Further study, and experimentation is needed to assess suitability of this approach.

4. OPEN QUESTIONS AND CONCLUSIONS

Quite sincerely, at the moment it is possible to implement only agent development part of a test-bed. Tools and methods for analysing time sensitive behaviour of multi-agent systems are still under development. The progress in this domain is considered in other papers. Therefore the test-bed development is a typical case of collaborative problem solving (pair programming) where the progress in test-bed fosters progress in theoretical development of analysis methods, and *visa versa*. So the test-bed development is by definition a non-terminating project, successful implementation of its first stage and experiments on it will inspire further research that will lead to new development stages, etc. Nevertheless, even a non-terminating activity should have interim goals, and checkpoints where the achievements are assessed and further development is decided.

The near future (the first stage) of the test-bed development is seen in three steps:

- Implementation of the shell and agent development tools, as described above
- Theoretical study of models of interactive computations with outcome related to formal analysis methods of respective multi-agent systems
- Implementation of timing analysis tool for the test-bed.

The on-line assessment tools, generation of measurement and reference data required by the assessment methods, the resulting change of time behaviour of the target system provide at the moment more questions than answers.

ACKNOWLEDGMENT

The partial financial support provided by ETF grant 4860 and grants no. 0140237s98, 0250556s98 from Estonian Ministry of Education is acknowledged.

REFERENCES

Bailin S.C. and Truszkowski W. (2001) "Ontology Negotiation: A Dynamic Approach to Substantive Communication between Agents", Proc. 5th World Multi-conference on Systemics, Cybernetics and Informatics, **vol. III**, 505-510

Caspi P. and Halbwachs N. (1986) "A Functional Model for Describing and Reasoning about Time Behaviour of Computing Systems", *Acta Informatica*, ***vol.22***, *595-627*

Evans R.P. (2001) "Design as Discovery. Ten System Design Tenets", Proc. 5th World Multi-conference on Systemics, Cybernetics and Informatics, **vol. XI**, 546-551

Ferber J. (1999) *"Multi-Agent Systems"*, Addison-Wesley, 509 pp

FIPA http://www.fipa.org

Fricke S., Bsufka K., Keiser J., Schmidt T., Sesseler R., and Albayrak S (2001) "Agent-based telematic services and telecom applications", *Communications of the ACM*, **vol.44**, no.4, 43-48

Goldin D. and Keil D. (2001) *"Interaction, Evolution, and Intelligence"*

JADE http://sharon.cselt.it/projects/jade/
Jennings N.R. (2001) "An agent-based approach for building complex software systems" *Communications of the ACM*, **vol.44**, No.4, pp.35-41
MadKit http://www.madkit.org
Milner R. (1999) *"Communicating and Mobile Systems: The PI-calculus"*, Cambridge University Press
Motus L. and Meriste M (2001) "Towards self-organising Time-sensitive control system's software", Proc. IFAC Conference on New Technologies in Computer Control, November 2001, Hong Kong, 236-241
Motus L. and Naks T. (2001) "Time models as used in Q-model and suggested for RT UML". 5th World Multi-Conference on Systemics, Cybernetics and Informatics, **vol. XI**, 467-472
Motus L (1995) "Timing problems and their handling at system integration", in S.G.Tsafestas and H.B.Verbruggen (eds) *Artificial Intelligence in Industrial Decision Making, Control and Automation*, Kluwer Publ., pp.67-88
Motus L. and Rodd M.G. (1994) *"Timing analysis of real-time software"*, Pergamon/Elsevier
Naks T. and Motus L. (2001) "Handling Timing in a Time-critical Reasoning System – a case study", *Annual Reviews in Control*, **vol.25**, 157-168
Rosenschein J.S. and Zlotkin G. (1998) *"Rules of encounter"*, The MIT Press, 229 pp.
Simon H.A. (1996) *"The Science of the Artificial"*, The MIT Press, 231 pp
Stone P. (2000) *"Layered Learning in Multi-agent systems"*, The MIT Press, 284 pp
Tsypkin Ya. (1971) " Adaptation and Learning in Automatic Systems", *Mathematics in Science and Engineering Series*, **vol.73**, Academic Press
Wegner P. (1997) "Why Interaction is More Powerful than Algorithms", *Communications of ACM* **vol.40**, No.5, pp.80-91
Wegner P. and Goldin D. (1999) "Co-inductive Models of Finite Computing Agents", *Electronic Notes in Theoretical Computer Science*, **vol.19**, www.elsevier.nl/locate/entcs
Wooldridge M. (2000) "On the Sources of Complexity in Agent Design", *Applied Artificial Intelligence*, **vol. 14**, no.7, 623-644