On Models for Time-Sensitive Interactive Computing

Merik Meriste [1] and Leo Motus[2]

[1] University of Tartu, Estonia
merik.meriste@ut.ee
[2] Tallinn Technical University, Estonia,
leo.motus@dcc.ttu.ee

Abstract. Agent-based paradigm is increasingly applied to building computing systems where conventionally latent requirements, e.g. time-sensitivity of data and event validity, and/or truth-values of predicates, timeliness of communication, and others become essential for correct functioning of systems. In many such cases empirical demonstration of expected behaviour is not sufficient, formal verification of certain properties becomes desirable. This assumes that interaction-based models of computing are to be enhanced by introducing sufficiently sophisticated time. Such enhancement is not too simple since time has been abstracted away from models of computing during the evolution of computer science. This paper introduces some preliminary ideas for developing time-sensitive interaction-based models of computations.

1 Introduction

The interaction-based models of computations have emerged from three independent research domains. Computer science and software engineering have reached the interaction-based models when developing methods for formal program verification (Milner (1980), Milner (1999)), and developing theoretical foundations for object-oriented programs and for programming in the large (Wegner (1997)). Interaction-based models of computations have attracted attention in distributed artificial intelligence due to the evolution of agent and multi-agent paradigm (see for instance Ferber (1999) and Wooldridge (2000b)). Many applications of real-time software are known for high dependency requirements. This has lead to studies that, departing from specific timing requirements of real-time software have lead to interaction-based models of computation (see, for instance Quirk (1977), Motus (1995)), applicable from the specification stage of system development. The latter models are not quite comparable with the conventional computer science models, because they focus on timing analysis in a component-based system.

Interaction-based models of computing exceed algorithmic models in formal power (see for details the notion of persistent Turing machines in Wegner (1997), Wegner and Goldin (1999)). For instance, persistent Turing machine can model any discrete-state computing agent that sequentially interacts with its environment. As a rule those models describe indefinitely on-going computations on a set of persistent Turing machines, whereas the interaction may depend on the pre-history of computations. Typical practical examples are multi-agent systems, real-time systems (e.g. systems

P.M.A. Sloot et al. (Eds.): ICCS 2002, LNCS 2329, pp. 156–165, 2002.

where computer is directly interacting with real-world processes – influencing thus behaviour of real-world processes and/or is being controlled by these processes, human-machine collaborative decision-making systems, and others.

Need for time-sensitive models of computations stems from specific problems in application domains. Computer control systems, increasing number of communication systems, multi-media, many distributed artificial intelligence and agents' applications naturally lead to time-constraint, concurrent software that assumes time-sensitive models of computations for its verification. Conventional computer science and software engineering have (unfortunately) successfully eliminated time (except as a trivial parameter for ordering events) from theories and tools related to program development. Most of the conventional theoretical thinking in computer science is based on an (not very realistic) assumption of completely known causal relations. Natural consequence of this assumption is that interest to time arises only at the physical design and/or implementation stages of software (e.g. scheduling) – and even then it is based on trivial performance type measures of execution time and execution frequency. Timeliness of interactions and validity of used and/or produced data is of no interest and cannot be evaluated by the existing models of computations.

Incomplete knowledge of causal relations is usual in time-constraint concurrent software and also in many agent-based applications. Software operation in such systems is, to a large extent, based on estimates, beliefs and/or deliberate approximations of causal relations – just to enable the use of conventional computing methods. Quite often causal relations are approximated by time constraints – just for the sake of economic considerations, or to enable the use of existing (too complex) knowledge, or to avoid paradoxes. For example a paradoxical situation may occur in continuous time when a causal relation becomes questionable if the time interval between the cause and effect reduces infinitely.

Another unavoidable feature of real-time and agent-based software is its indefinitely on-going nature of execution and persistency of interactions' influence on the future behaviour. Wegner and Goldin (1999) state that models of computations must be adequately enriched in order to model persistence of computing agent state and on-going interaction with a possibly incomputable environment. An example from conventional computer science illustrates what may happen if the models are not properly expressive and analysable – consider a non-terminating program as a model of indefinitely on-going program. Because of too abstract notion (i.e. non-termination) that hides the intrinsic features of the on-going computations, the whole topic of non-termination was left out of the mainstream theory for a long time (as not easily tractable and therefore not interesting). Just remember that termination is often a precondition to verifiability of a program. This is a subjective obstacle to verification of such programs. The objective obstacle is that the usually applied first-order predicate calculus cannot describe the functioning of an interaction-based non-terminating program – one needs higher order predicate calculus for that (see for instance, a claim in Wegner and Goldin (1999), an example in Lorents et alii (1986)). A statement from Goldin and Keil (2001) partly explains the essence why higher-order logic is needed – no sequence of preordained steps (or series of interactions) can model the multiple-stream encounters between multi-interacting evolved agents and their environments.

Serious confusion also rises when one tries to explain the essence of a non-terminating program to the end-user (who is not a specialist in computer science). Besides, the notion of "non-terminating program" is an approximate description that hides inner details of the program. More precise, and also more acceptable for non-computer specialists is the description of a "non-terminating program" as a set of interacting, repeatedly activated, terminating programs (Quirk (1997)). This concept easily enables to introduce sufficiently sophisticated time and naturally leads to formal verification of timing properties in the early life-cycle stages – e.g. specification and logical design (Lorents et alii (1986) and Motus (1995)).

Conceptually closer to the traditional notion of "non-terminating program" is the departing point for timed process algebra studied by Caspi and Halbwachs (1986)) as an approach to timing analysis of software. This approach encountered serious theoretical problems due to use of continuous time that has hindered its practical application.

Interaction-based computing is an input/output stream processing view on computations as opposed to algorithmic computations where an input string is processed, output string is generated and then the algorithm terminates. The input/output stream processing emphasises the persistent nature of interactive computing. Characteristic to interactive computing is the phenomenon of emergent behaviour (Simon, 1970) that can be observed in multi-agent systems, real-time systems, human organizations, and in many other applications. In such systems agent's learning implies adaptation and, adaptation in turn implies interaction. Learning is an example of history-dependent behaviour that cannot be adequately modelled within an algorithmic (i.e. without history) model of computations neither in an order-of-events setting (Goldin and Keil, 2001).

Besides the emergent behaviour, interactive computations have the following features that are not present in algorithmic computations: (1) stream input-output, (2) interleaving of inputs and outputs during computations, (3) history-dependent behaviour of the computing agents. Causal relations built into the computing system define admissible permutations in interleaving input and output streams. When the causal relations are not completely known one cannot define precisely which of the occurring permutations are admissible. Another difficulty is related to exceptional permutations potentially occurring when one implements forced (Wegner and Goldin 1999 use "true") concurrency that occurs in the case of multi-stream interaction machines or forced (and too high) frequency of input stream for sequential interaction machines. Such phenomena may occur, for instance, as disturbances even in algorithmic computations on multi-processors with pipeline or matrix architecture.

Such anomalies, including those potentially caused by incomplete knowledge of causal relations, can be avoided by introducing suitable time constraints. However, this assumes that instead of an abstract notion of non-terminating program is considered a more precise inner structure – e.g. persistent Turing machine (suggested by Wegner and Goldin (1999)), or a set of interacting, countable number of times activated, terminating programs (suggested by Quirk (1977), Motus (1995)).

This paper focuses on preparatory aspects of developing time-sensitive interaction-based model of computations – selection of suitable candidates for comparative study of interaction-based models of computations, and on discussing the sufficient complexity of time to be used in the model of computations. Most of the future work is planned to be theoretical. However, occasional experimental checks of the theory are important. In addition to the existing tool that implements an interaction-based model of computations (for timing analysis of systems behaviour), the authors have started design and implementation of another "by-product" – a test-bed for assessment and analysis of development and evolution of multi-agent systems. The testing tools and experiments will be considered in a separate paper.

2 Interactive Models of Computations

There is no widely accepted model of interactive computations for the time being. Instead, several concepts and models have been suggested. For instance, the calculus of communicating systems and pi-calculus (Milner 1980, Milner 1999), Q-model (Motus and Rodd 1994); temporal logic of reactive and concurrent systems (Manna and Pnueli, 1991), logic of rational agents (Wooldridge, 2000a), input/output automata (Lunch and Tuttle, 1989), abstract state machines (Gurevich, 1993), attributed automata (Meriste and Penjam 1995), interaction machines (Wegner 1997).

In the pragmatic world of real-time systems the mainstream research is still trying to patch holes in the algorithmic approach. Models of interactive computing are attracting attention slowly, in spite of the successful start in the 1980-es and some interesting practical results obtained recently. For instance, consider a paper on application of a time-sensitive model of interactive computations (Naks and Motus (2001)), and suggestion for a sufficiently sophisticated time-model to be applied for RT UML project (Motus and Naks (2001)).

Respective studies with models of interactive computations in the agents' community and object-oriented community have progressed faster, see for instance, Milner (1999), Wegner (1997), Wegner and Goldin (1999), Wooldridge (2000a), Jennings (2001). Respecting the computer science traditions, however, time has been abstracted away from the models of interactive computations resulting from the latter communities.

Modelling the stream-based behaviour of a computing system has been the key problem in those communities. Behaviour usually denotes a set of I/O streams that obey certain pre-defined ordering discipline. In many cases the ordering discipline are defined by setting only a few constraints on certain interactions. This enables to cope with potentially non-countable number of different behaviours generated by a persistent interactive machine (with countable number of interactions). It is suggested that by changing mathematical framework from induction to co-induction, from least fixed point to greatest fixed point, the problem of model expressiveness can be resolved (see for details Wegner and Goldin (1999), Goldin and Keil (2001).

The analysing capability of models is also important, although somewhat neglected at this development phase. In many cases the analysing capability can be reduced to

the question of defining admissible behaviours, distinguishing equivalent sets of behaviours from each other, and demonstration that only admissible behaviours are generated (or that no prohibited behaviour is generated). Behaviour is defined by order imposed on elements in the I/O stream, usually by precedence relation. Concurrent occurrence of stream elements is often demonstrated by interleaving streams.

Difficulties may appear when considering the case of incompletely known causal relations between the agents (components, etc), or the case when causal relations are completely known, but the use of this knowledge is computationally too expensive (e.g. takes more computing power than one has). This usually implies that I/O stream elements cannot be ordered *a priori*. On the outskirts of computer science computing systems seldom function fully alone, independently and autonomously (i.e. normally they interact directly with the surrounding world). In such cases the admissible behaviours are often dynamically defined by responses from a natural (or artificial) world processes that are (at least partially) out of the control of the computing system. Remember, for instance, the case of emergent behaviours (Goldin and Keil (2001)).

In many applications not only the logical order of stream elements is important for defining behaviours. In time-critical applications, for instance, the length of time interval between the respective elements of the I/O stream may be decisive for behaviour classification. This implies the introduction of a metric time in addition to logical time (ordering). One metric time per system may not be sufficient. In multi agent systems and in real-time systems each agent (component, subsystem) may have its own time system, imposed by its own immediate environment. Those separate time systems are often quite different from each other and can only approximately be projected to a single system timeline. Nevertheless, for analysing properties of inter-agent (inter-component) interactions, one needs all those different time systems – approximate system's time may not be sufficient for analysing salient time properties of interactions (Motus (1995).

The way ahead. The authors of this paper believe that interaction based computing systems are increasingly applied to time-sensitive applications. Such applications provide essentially incomplete information about their (potentially dynamically changing) intrinsic details and interactions – e.g. goal functions, characteristics of components, inner structure, influence from the environment, etc. Formal analysing tools are not yet available, although their development is supported by OMG initiatives in developing real-time extensions to UML, and ongoing research in applying agent-based paradigm (e.g. distributed computer control systems, mobile robotics, banking). Nevertheless, the everyday practice is still relying on "trial and error" method.

To partially suppress a new round of "trial and error" activities it makes sense to try and merge the favourably advanced theoretical ideas from all the three involved source domains – computer science, real-time systems, and multi-agent systems – and develop a time-sensitive model of interactive computations.

Based on the above discussion the major question (in the context of this paper) in models of interactive computations seems to be "how to improve their analysing capabilities in the case of incomplete information, emerging behaviours and quantitative time-sensitive behaviours". Previous theoretical and experimental

experience in the domains of real-time systems and time-critical reasoning lead the authors to the following

Working hypothesis: *Introduction of reasonably sophisticated time system into the description of model of interactive computations enables to reduce indeterminacy of behaviours generated by the model by approximating incompletely known causal relations, catering for quantitative time constraints, and ordering interactions in time even on multi-stream interaction machines.*

Hence, co-inductive definitions permit to consider the space of all computational processes as a well-defined set, even if the input streams are generated dynamically (Goldin and Keil (2001)). The sufficiently sophisticated time system adds the ability to introduce adequate ordering disciplines – required for analysis of system properties. It is stressed that time system can also cater for forced concurrency created (pure computational) problems in multiprocessors with pipeline and/or matrix architectures – by introducing additional ordering constraints.

2.1 Criteria for Qualifying as a Candidate for Time-Sensitive Model of Interactive Computations

The first problem is to fix (preliminary) criteria that limit the number of candidate models to be considered in more details for developing time-sensitive model of interactive computations. The qualified models are then studied further to understand their specific and common properties, one or two candidates will eventually be chosen.

The first criterion emphasises that essential requirements of multi-agents and real-time systems are to be met by candidate models. It seems reasonable to depart from the characteristic properties for multi-stream interaction machines stated by Wegner and Goldin (1999):

- Capability to handle true concurrency (called forced concurrency in Motus and Rodd (1994))
- Higher than first order-logic is required to describe their functioning.

By pure coincidence the same features characterise the Q-model and related weak second order predicate calculus discussed in Motus and Rodd (1994) and Lorents et alii (1986).

The second criterion facilitates comparison of candidate models by partition them into subgroups. During the preliminary analysis the candidates are subjectively classified based on the (at least seemingly) prevailing methodology, e.g.:

- Process algebras represented, for instance, by CCS and PI-calculus (Milner (1980) and Milner (1999)); history transformer suggested by Caspi and Halbwachs (1986) is also a good candidate
- State machines, represented, for instance, by interaction machines by Wegner (1997), attributed automata (Meriste and Penjam (1995), statecharts (Harel (1987))

- Logical framework based models, represented by temporal logic (Manna and Pnueli (1991), logic of rational agents' Wooldridge (2000a), weak second order predicate calculus with time (Lorents et alii (1986)).
- Other (pragmatic) models represented, for instance, by Wooldridge (2000b), Harel (2001), Motus and Rodd (1994), and others.

Thorough research of the qualified models will hopefully reveal one or two suitable candidates for further elaboration to a time-sensitive model of interactive computations by introducing the required system of time-features. The resulting time-sensitive model is to be used as one of the models for describing agents and systems of agents in a concurrently developed test-bed – the experimental side of this theoretical study. The respective methods developed for formal analysis of systems properties form the basis of the assessment tool to be developed and used in the test-bed.

3 On concepts of Time as Required in Time-Sensitive Multi-agent Systems

Different authors have used the notions of "time", "time properties", "timing analysis", and "time models" in a variety of contexts and meanings. Some comments follow in order to reduce potential confusion and to explain the essence of the time as used in models of computations. A large set of disparate views on the role of time in computing is partly the reason for the excessive variety of terms, beliefs and understandings. For instance, implementation stage (e.g. scheduling theory) needs different time notion than specification and design stages. In the latter stages the time constraints are derived from the environmental and user requirements and imposed on components and their interaction. In the implementation stage (scheduling) all those time constraints are projected on one single time-line and checked for satisfaction – if the system is schedulable. If not a new error and trial iteration is started. This is where sophisticated time and formal timing analysis is of real use (see, pre-run-time scheduling (Xu and Parnas, (2000)); timing analysis at specification stage, (Motus and Rodd (1994))).

It has been demonstrated in Motus and Rodd (1994) that a conventionally used single time variable per system can be sufficient only for assessing performance type of timing properties (usually the focus of scheduling methods). Instead, a rather sophisticated time model should be used in order to handle also the truly salient types of timing properties – time-correctness of events and data, and time correctness of interactions between parts of an agent, agents, coalitions of agents, etc.

A system is compiled from components (reused software pieces, COTS, agents, subsystems, etc) whose dynamic properties are important for reaching the goal of the component, and the system as a whole. A component is loosely connected to the other components (by exchanging messages), and may manage its own connections with the environment. Due to the remarkable potential autonomy of components each component may have its one, independent of the other components, time system. Usual members of a component's time system are, for instance:

- thermodynamic time for accounting the overall progress of the component and its environment,
- reversible time for potential handling of faults and exceptional cases inside of the component, and
- a set of relative times for describing and analysing interactions with the other components.

At the first glance, such time model looks extremely sophisticated. In reality it boils down to intuitively well understandable for a non-computer specialist, and easily specifiable model that enables automatic verification of many timing properties (see, Motus and Rodd (1994), Motus (1995)) and simulation sessions on an automatically generated prototype already at the specification stage. Automatic verification means that the user is saved from theoretical nuances required for statement theorems about common timing problems – e.g. non-violation of imposed timing constraints during an interaction. Application-specific timing properties still need specific handling (e.g. statement and proof of theorems) as usual.

4 Conclusions

The number of time-sensitive applications of interactive computing is increasing fast. At the same time practical development is, especially at the early development stages, still largely based on trial and error method. This paper discusses premises for developing a time-sensitive model of interactive computations that would enhance theoretical understanding of what goes on at the early stages of systems development and assist in early detection and elimination of inconsistencies – including timing inconsistencies.

The paper discusses the necessity of time in computing systems and states a working hypothesis that reasonably sophisticated time system could substantially assist in reducing indeterminacy introduced by incompletely known causal relations. Time could also facilitate the handling of constraints imposed on computational processes and their interactions by direct communication of the computing system with its dynamically changing environment.

The paper also sketches a way towards systematic development of the time-sensitive model of interactive computations, starting form the variety of existing models and also considering the experience obtained in formal timing analysis of real-time software.

It is acknowledged that true stable progress in developing a theoretical model can be guaranteed only by merging the theoretical study with experiments. Therefore, the authors suggest that, as a part of future activities, building a test-bed is to be started. Such a test-bed would serve as a discovery system for agent-related knowledge, in addition to being just an assessment environment for theoretical results. For instance, the test-bed would enable (based on the model that is to be developed) explicit experimental study of time sensitivity of parts of agents (related to communication, learning, and adaptation), and agents' interactions.

Acknowledgment

The partial financial support provided by ETF grant 4860 and grants nr 0140237s98, 0250556s98 from Estonian Ministry of Education is acknowledged.

References

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

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

Goldin D. and Keil D. (2001) "Interaction, Evolution, and Intelligence", Congress on Evolutionary Computation, Korea, May, 2001. http://www.cs.umb.edu/~dqg/papers/cec01.doc

Gurevich Y. (1993). Evolving algebras 1993: Lipari guide. *In* Borger, Ed., *Specification and validation methods,* 1995, pp. 231-243.

Harel D. (1987) "State-charts: a visual formalism for complex systems", Science of Computer Programming, **Vol.8** (1987), p.231

Harel D. (2001) "From play-in scenarios to code: an achievable dream" Computer, **Vol.34**, No.1, 53-60

Jennings N.R. (2001) "An agent-based approach for building complex software systems" *Communications of the ACM*, **Vol.44**, No.4, pp.35-41

Lorents P., Motus L., Tekko J. (1986) "A language and a calculus for distributed computer control systems description and analysis", Proc. on Software for Computer Control, Pergamon/Elsevier, 159-166

Lynch N. A. and Tuttle. M. R. (1989) An introduction to input/output automata. *CWI Quarterly* 2(3) (September 1989), pp. 219-246.

Manna Z. and Pnueli A. (1991) "The temporal logic of Reactive and Concurrent Systems", Springer Verlag

Meriste M. and Penjam J. (1995) "Attributed Models of Computing". Proc. of the Estonian Academy of Sciences. Engineering, **Vol.1**, No.2, pp.139-157

Milner R.A. (1980) "A Calculus of Communicating Systems" LNCS, **Vol.92**, Springer Verlag, 171p

Milner R.A. (1993) "Elements of Interaction". Turing Award Lecture, *Communications of the ACM*, **Vol.36**, No.1, pp.78-89

Milner R. (1999) "*Communicating and Mobile Systems: The PI-calculus*", Cambridge University Press, 161p

Motus L. and Naks T. (2001) "Time models as used in Q-model and suggested for RT UML". Proc. 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

Quirk W. and Gilbert (1977) "The formal specification of the requirements of complex real-time systems", AERE, Harwell, rep. No. 8602

Wegner P. (1995) "Interaction as a Basis for Empirical Computer Science", ACM Computing Surveys, **Vol.27**, No.5, pp.80-91

Wegner P. (1997) "Why Interaction is More Powerful than Algorithms", *Comm. 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. (2000a) "Reasoning about rational agents" MIT Press, 227 p

Wooldridge M. (2000b) "On the Sources of Complexity in Agent Design", *Applied Artificial Intelligence*, **Vol.14**, No.7, 623-644

Xu J. and Parnas D.L. (2000) "Priority Scheduling versus Pre-run-time Scheduling", The International Journal of Time-critical Computing Systems, **vol.18**, no.1, 7-23