

FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS

FIPA Modeling Area: Temporal Constraints

Document title	FIPA Modeling Area: Temporal Constraints		
Document number	TBA	Document source	FIPA Modelling TC
Document status	Proposed	Date of this status	2003/04/18
Supersedes	None		
Contact	rlevy@i-a-i.com		
Change history			

Working Draft

Version 1.0 - 2003-04-18 13:34EST

© 2003 Foundation for Intelligent Physical Agents - <http://www.fipa.org/>

Geneva, Switzerland

Notice

Use of the technologies described in this specification may infringe patents, copyrights or other intellectual property rights of FIPA Members and non-members. Nothing in this specification should be construed as granting permission to use any of the technologies described. Anyone planning to make use of technology covered by the intellectual property rights of others should first obtain permission from the holder(s) of the rights. FIPA strongly encourages anyone implementing any part of this specification to determine first whether part(s) sought to be implemented are covered by the intellectual property of others, and, if so, to obtain appropriate licenses or other permission from the holder(s) of such intellectual property prior to implementation. This specification is subject to change without notice. Neither FIPA nor any of its Members accept any responsibility whatsoever for damages or liability, direct or consequential, which may result from the use of this specification.

Foreword

The Foundation for Intelligent Physical Agents (FIPA) is an international organization that is dedicated to promoting the industry of intelligent agents by openly developing specifications supporting interoperability among agents and agent-based applications. This occurs through open collaboration among its member organizations, which are companies and universities that are active in the field of agents. FIPA makes the results of its activities available to all interested parties and intends to contribute its results to the appropriate formal standards bodies.

The members of FIPA are individually and collectively committed to open competition in the development of agent-based applications, services and equipment. Membership in FIPA is open to any corporation and individual firm, partnership, governmental body or international organization without restriction. In particular, members are not bound to implement or use specific agent-based standards, recommendations and FIPA specifications by virtue of their participation in FIPA.

The FIPA specifications are developed through direct involvement of the FIPA membership. The status of a specification can be either Preliminary, Experimental, Standard, Deprecated or Obsolete. More detail about the process of specification may be found in the FIPA Procedures for Technical Work. A complete overview of the FIPA specifications and their current status may be found in the FIPA List of Specifications. A list of terms and abbreviations used in the FIPA specifications may be found in the FIPA Glossary.

FIPA is a non-profit association registered in Geneva, Switzerland. As of January 2000, the 56 members of FIPA represented 17 countries worldwide. Further information about FIPA as an organization, membership information, FIPA specifications and upcoming meetings may be found at <http://www.fipa.org/>.

Contents

- 1 Scope..... 1
 - 1.1 Abstract..... 1
 - 1.2 Keywords 1
 - 1.3 Document participants..... 1
- 2 Overview Of Temporal Constraints..... 2
 - 2.1 Time domains..... 2
 - 2.1.1 Types of time domains..... 2
 - 2.1.2 Dependencies between time domains..... 3
 - 2.2 Temporal Constraints..... 3
 - 2.2.1 Types of temporal Constraints..... 4
- 3 Diagrams..... 4
 - 3.1 Modelling of time domains 4
 - 3.2 Modelling Multiple Domains and their Interdependencies..... 5
 - 3.3 Modelling of Temporal Constraints..... 5
 - 3.3.1 Delays 5
 - 3.3.2 Timers..... 6
 - 3.3.3 Time-out 7
 - 3.3.4 Composed Constraints..... 8
- 4 Example..... 8
- 5 References 8

1 Scope

1.1 Abstract

In this paper we focus on presenting the several temporal conditionals by which agent based systems may be constrained. We define such constrains and demonstrate how they can be modelled using extensions to the current diagrams used to describe multi-agent systems.

1.2 Keywords

Time constraints, time domains, real-time, emulation, delay, timer, and synchronization

1.3 Document participants

Renato Levy, Intelligent Automation, Inc.

<Your name goes here>

2 Overview Of Temporal Constraints

Multiagent systems may necessitate that temporal constraints be reflected on their modelling. These constraints may refer to limits in the time allowed between two interactions (delay) or to more complex dependencies between its agents or roles. Furthermore since agents are independent and frequently are used in simulation environments, these temporal constraints are not necessarily within the same scale/format (time domain), the exact nature of the time domains in the system and their interdependencies have also to be modelled with the description of multi-agent systems.

2.1 Time domains

A time domain is a set of rules and definitions that will control the passage of time for a system. Each time domain has its current value of time and the exact set of rules that control the way this value evolve. The very notion of time prevents the current value of the time domain to flow backwards, or in other words, for a given execution run, the next value of a time domain T is not smaller than any of its predecessors.

$$(1) \quad T_{(i+1)} \geq T_{(i)}$$

2.1.1 Types of time domains

Domain nature:

Continuous → A continuous time domain implies a constant passage of time. In these time domains the current time value varies continuously and in any two instants the current time value is always different. This is how humans perceive the actual passage of time. Continuous Time domains have very little applicability in computer science solutions since in practical terms; the domain has to be sampled with some level of frequency. Continuous time domains can only be applied with the proportional rule of propagation (see below), and it is usually based against the real time.

Discrete → A Discrete time domain partition the passage of time in specific intervals. Any event that occurs within the interval is observed to have occurred at its limits. The granularity of a discrete time domain is the size of such intervals (1second, 1 minute, ...) and represents the minimum increment that the current time value can be modified. In practical terms, continuous time domains are modelled in discrete domains by selecting a proper granularity for the application.

Rule of propagation:

Proportional → Proportional time domains use the actual passage of time as a reference to update the current time value. These domains associate the current value of time linearly with the actual sensed time interval since the execution started.

$$(2) \quad T_p = _t + T_0, \text{ where } _ \text{ is the proportionality factor,}$$

$$t \text{ is the actual time interval,}$$

$$\text{and } T_0 \text{ is the initial value of the time domain}$$

When the proportional time domain is discrete, the value of T_p has to conform with the granularity of the domain.

Periodic → This type of propagation is applied to discrete time domains. Under this rule of propagation the current value of time domain is periodically incremented by a constant value. The actual increment value is proportional to he granularity of the time domain.

$$(3) \quad T_f(i+1) = T_f(i) + _g, \text{ where } _ \text{ is the proportionality factor,}$$

$$\text{and } g \text{ is the granularity of the time domain, } g \geq 1 \text{ and } g \in \mathbf{N}^1$$

¹ \mathbf{N} is the set of natural numbers.

Next Event \rightarrow The next event propagation rule is used in discrete time domains. Under the next event rule the current value of the time domain is not updated until an event to update happens. When the trigger event is sensed the current value of the time domain is updated to the next expected event value. Simulations that use next event time domains avoid using computational cycles to represent unimportant moments of the system.

$$(4) T_n(i+1) = t_e, \text{ where } t_e \text{ is the time of the next expected event and } t_e > T_n(i)$$

2.1.2 Dependencies between time domains

In multiple time domain systems, some domains might be dependent on others, in a sense that their rules of propagation are not completely free to operate. To illustrate this effect let's consider two agents that use their own clocks to coordinate their actions. In the first agent the clock indicates 2 minutes before noon, whereas for the other agent the clock is set to 5 minutes before noon. If the two agents plan to meet at noon precisely, the meeting will not occur, but if they agree to meet at noon and wait at most 10 minutes for the other agent, then they will meet, because although the instant values of time for each are different, the passage of time is constant in both time domains.

The following are the possible dependencies between time domains:

Synchronization \rightarrow Synchronized time domains mean that passage of time in one domain is bounded by the passage of time in another domain, and that whenever possible to its domain features the current values of such time domains should be equal. The two domains do NOT need to be of the same type and/or granularity. In case the granularity of the domain is different, the current value at the domain that is coarser is always less than one interval apart from the value in the finer domain. The values of such domains will be equal on multiples of the interval time.

$$(5) T_a = \square_1 g_a \text{ and } T_b = \square_2 g_b, \text{ then if } g_a < g_b \rightarrow \square_1 g_a \leq \square_2 g_b \leq (\square_1 + 1) g_a$$

$$(6) \text{ if } t = \square g_a g_b; \text{ then } T_a = \square_1 g_a = \square_2 g_b = T_b, \text{ where } \square, \square_1, \text{ and } \square_2, \text{ are in } \mathbf{N}^* \quad ^2$$

Offset \rightarrow Time Domains, which are offset with another, keep their current time value at a constant distance from each other. The example of the meeting of two agents with different clocks is an example of the offset dependency. Observe that the two time domains need NOT to be of the same type and/or granularity. The equations below describe the relationship between the time domains when the granularity of each domain is different for an offset of value J.

$$(7) T_a = \square_1 g_a \text{ and } T_b = \square_2 g_b, \text{ then if } g_a < g_b \rightarrow \square_1 g_a \leq \square_2 g_b + J \leq (\square_1 + 1) g_a$$

$$(8) \text{ if } t = \square g_a g_b; \text{ then } T_a = \square_1 g_a = \square_2 g_b + J = T_b, \text{ where } \square, \square_1, \text{ and } \square_2, \text{ are in } \mathbf{N}^*$$

Drift \rightarrow Two time domains will keep a drift dependency if the offset between the two time domains is a function of current value of one of them, or a third controlling time domain. As in the other dependencies, this condition is insensitive to the individual granularities. In a sense the two previous dependencies can be seen as degenerate cases of the Drift dependency. The equations governing the dependency for a drift function F and controlling time Domain T_c , are given below:

$$(9) T_a = \square_1 g_a \text{ and } T_b = \square_2 g_b, \text{ then if } g_a < g_b \rightarrow \square_1 g_a \leq \square_2 g_b + F(t_c) \leq (\square_1 + 1) g_a$$

$$(10) \text{ if } t = \square g_a g_b; \text{ then } T_a = \square_1 g_a = \square_2 g_b + F(t_c) = T_b, \text{ where } \square, \square_1, \text{ and } \square_2, \text{ are in } \mathbf{N}^*$$

2.2 Temporal Constraints

Temporal constraints are time related relationships that must be kept in the multi-agent system. Such relationships can be within the definition of a role's behaviour (vertical constraint) or between the cross section of the role's involved in a protocol (horizontal constraint). In this section we define the type of temporal constraints covered in this document.

² \mathbf{N}^* is the set of natural numbers without the zero

2.2.1 Types of temporal Constraints

Delay→ Delays can be vertical or horizontal constraints and indicate limits in time difference between two distinct events for the MAS system. Vertical delays are present on a single role and should be related to time values from a time domain visible to this role. Horizontal delays may involve more than one time-domain and domain visibility has to be guaranteed for role in delay’s definition.

Timer→ Timer constraints indicate the automatic issuing of a time related event after a pre-determined time has expired in reference to a visible time domain. Timers can also be used to indicate a repetitive task that occurs at specific time intervals. Timers are defined within a single time domain although it might be represented horizontally within this domain’s visibility.

Time-out→ The time-out constraint is a variation of timers that provide means for the system to pursue an alternative behaviour in case a pre-determined condition is not achieved within a certain period of time. Just like timers, time-out constraints are limited within the visibility of a single time-domain.

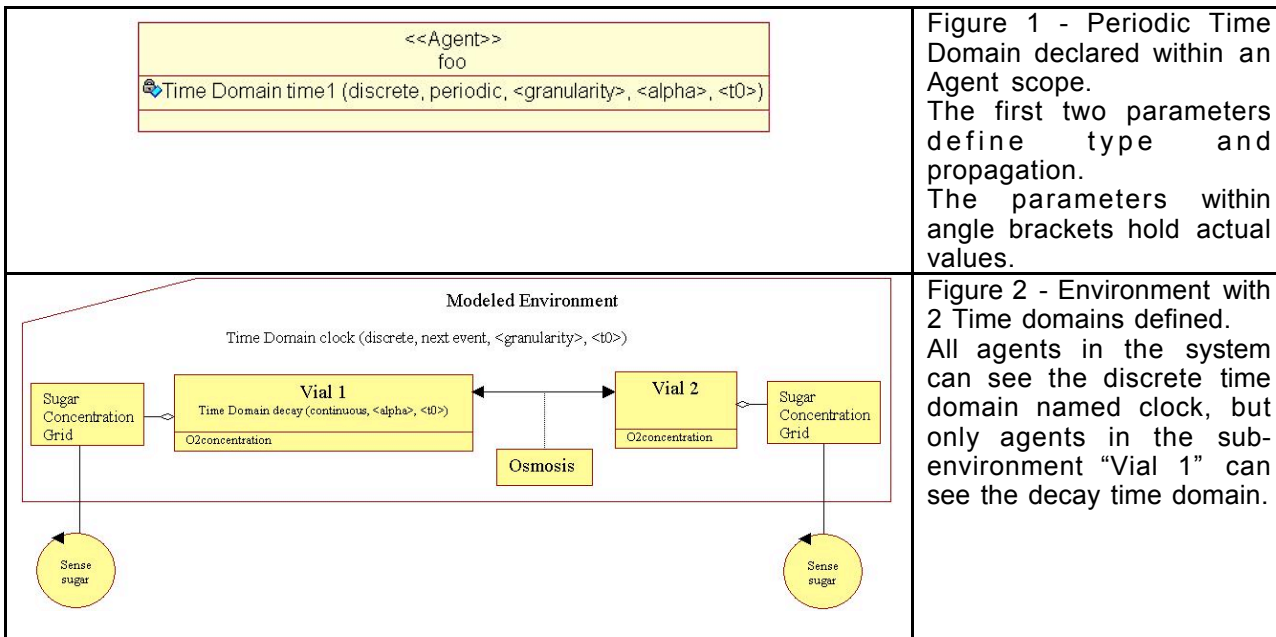
Composed Constraints→ Composed constraints are a relation between multiple time constraints, where each constraint may be related to a different time domain. All time domains involved should be visible at the role/agent/level in which the constraint is imposed.

3 Diagrams

In this section we will demonstrate the diagrams used to model time domains and temporal constraints.

3.1 Modelling of time domains

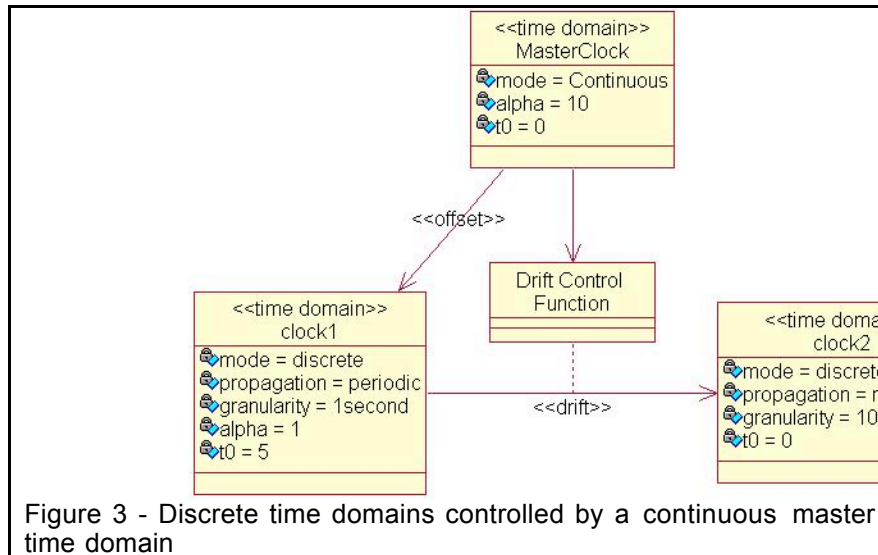
Modelling a time domain means to indicate its type and rule of propagation along with its parametric values. A time domain has to be modelled at the highest level that relates to its visibility. As an example is the time domain is restricted to an agent, then the correct place to define this domain is at the agent level, on the other hand if all members of a group share the visibility of this domain, then the time domain should be defined at the group level. The notion of time domain visibility in this definition is conceptual and therefore independent of platform implementations. Figures 1 and 2 indicate the modelling of time domain definitions on different levels of a multi-agent system.



Note: the actual final modelling format will be modified to comply with the final format of agent/group and environment modelling formats.

3.2 Modelling Multiple Domains and their Interdependencies

Multiple time domains of a system may be modelled using an extension of the class diagram with associations between time domains marked with the proper dependency schemes as defined in section 2.1.2. Alternatively one might indicate a controlling class to support the drift relationship between two time domains. Figure 3 indicates how this modelling can be achieved.



In figure 3, we demonstrate the relationship between three time domains. The master clock is a continuous clock that executes 10 times faster than a “real life” clock. Clock1 domain will track this clock discretely with a granularity of one second. Clock 2 is a discrete clock, which propagates from event to event, but since it is forced to drift around clock1, it is automatically bounded on how much in the future it can project events. Therefore even if there is no event scheduled for clock2, its value will at most drift to a value bounded by a function of the value of the master clock, applied to the instantaneous time value of clock1. Clock2 is also forced to observe its own granularity. The final equation for clock 2 can be defined as:

$$t_2 = \text{round} ((f(t_{\text{master}}) t_1) / 10) * 10; \text{ where } f \text{ is the drift control function}$$

and the granularity adjustment is considered to round off errors,
other options for adjustments would be: truncate, ceiling, floor,

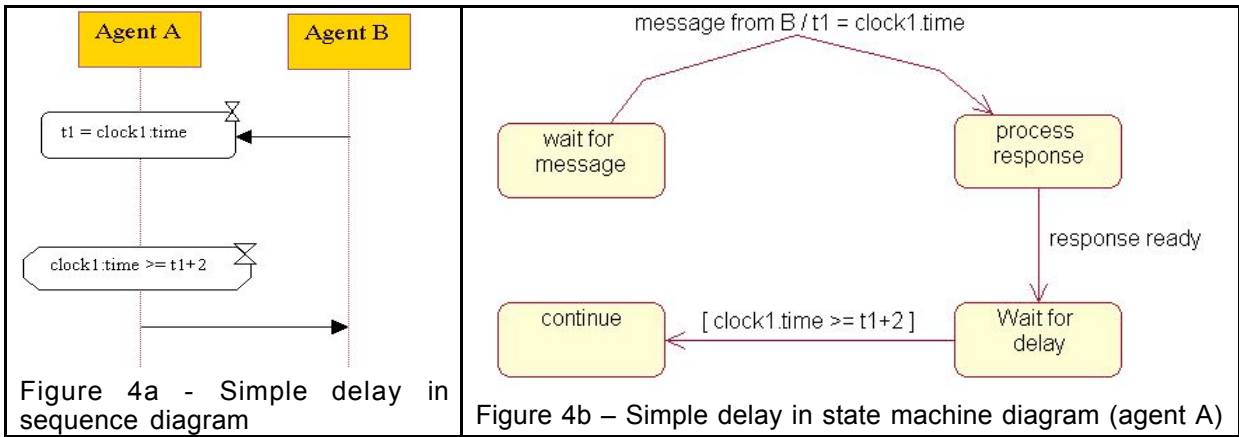
The proposed notation is quite rich and can translate complicated time domain relationships. This richness introduces yet the problem of guaranteeing that the notation does not introduce dualities of interpretation, which will force us to describe precisely the mainning of each relationship between time domains type and propagation rules.

3.3 Modelling of Temporal Constraints

In this section we will present how the temporal constraints defined may be represented in sequence and state-diagrams. Contributions for these representations in collaboration and activities diagrams are welcomed.

3.3.1 Delays

The first example of delay is for a vertical delay within the same agent/role for a single time domain. In the example in figure 4a, Agent A has to respond to a message from agent B, but not before two units for time domain clock1 have passed since the message from B was received. Figure 4b shows the same example as seen by agent A in a state diagram.



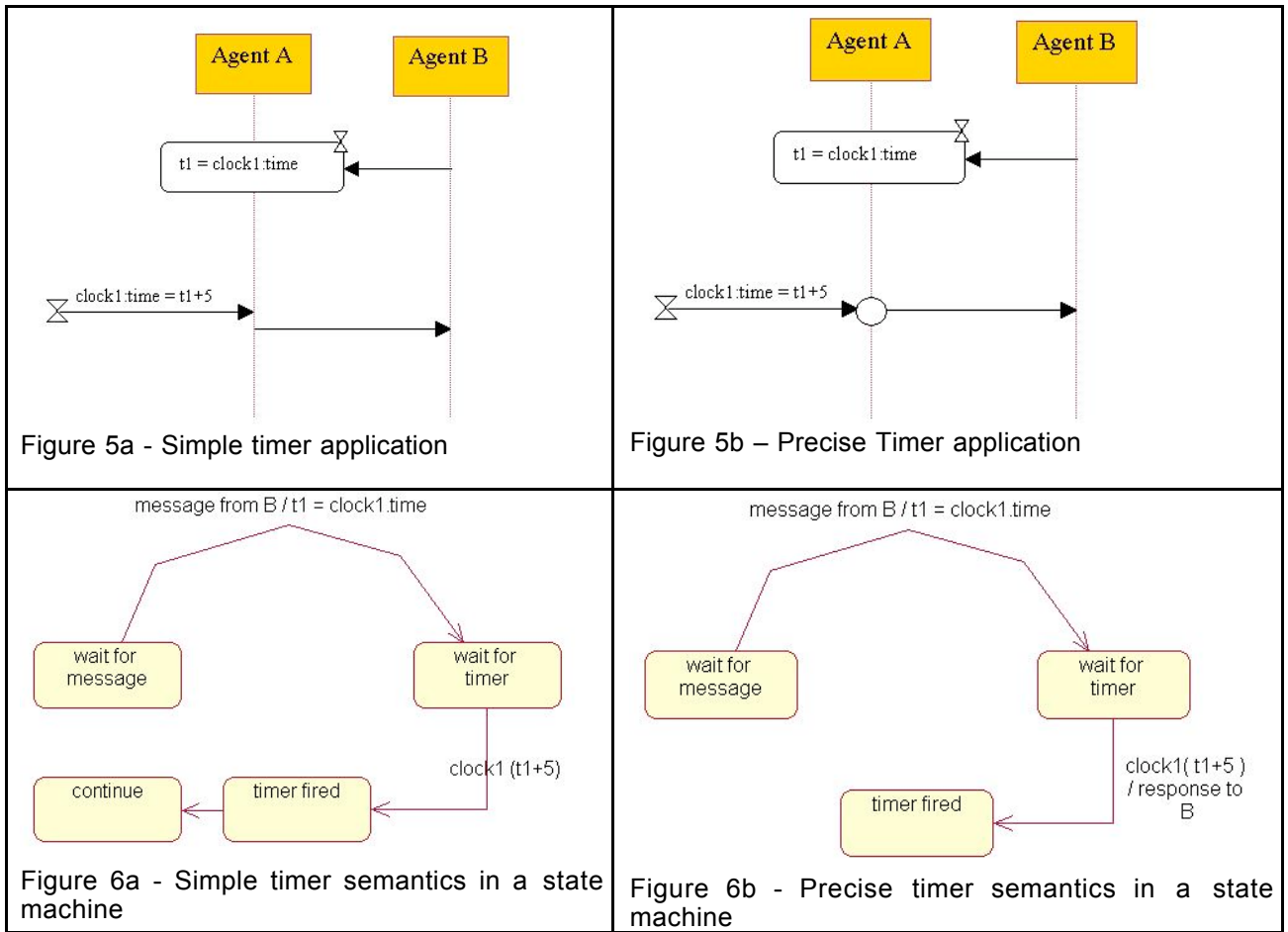
The simple delay example in the sequence diagram introduces two new concepts and two notation icons. The first concept introduced refers to the capture of the current value of a specified time domain. This concept is represented on the form *<time domain name>.time*. The second concept introduced is the time variable. The time variable is used in diagrams to register the value of a time domain, in order for it to be used in future calculations/logic. The time variable has no meaning or definition in the system been modelled and it is used only as an auxiliary tool. In the figure 4a, “t1” is a time variable.

The two new icons introduced for notation is the time assertion icon, noted by a round edge rectangle with a small sand-clock at the top right corner. The assertion icon is used to initialise a time variable. The second notation icon introduced is the actual time constraint icon, which is denoted by a decision hexagon with the clock sand at the top right corner. Inside the hexagon the designer introduces the constraint to be met by the system.

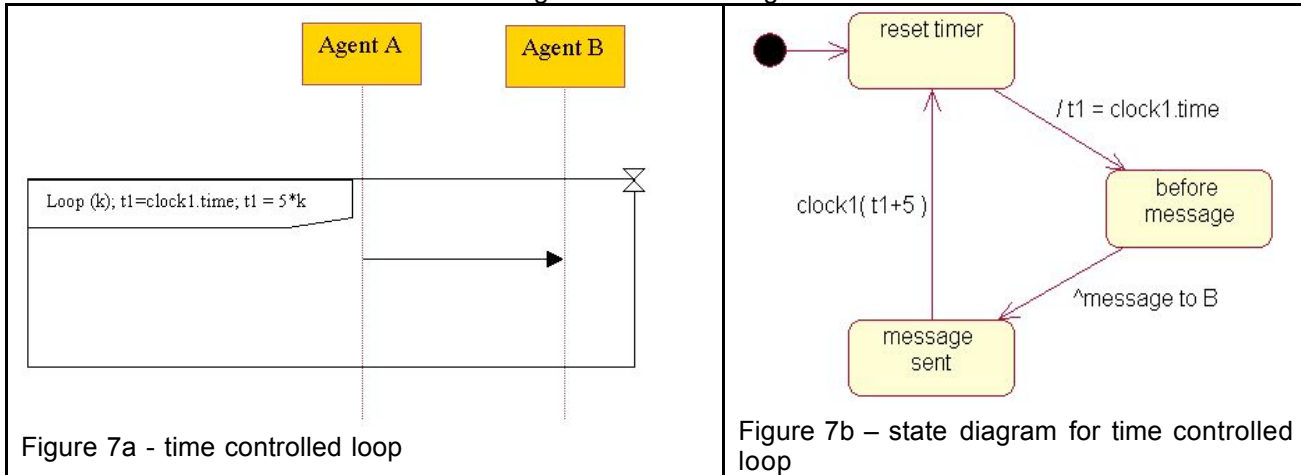
We can observe that no special notation is needed to represent the simple delay in the state diagram, with the exception of the introduction of the time variable (t1) and concept of current value of time domain which was introduced before for the sequence diagram.

3.3.2 Timers

A timer can be described by two moments: The moment of setting the timer and the moment when the timer fires. In figure 5a, we demonstrate how the notation for a timer can be introduced in the sequence diagram. Observe that the moment of setting the timer is actually the usage of a time variable (t1) to mark a moment in a time domain. The actual expiration of a time is shown by having a message come to the agent/role from the sand clock. In figure 5a, agent A respond to agent B anytime after the timer fires. A slightly different semantics is shown in figure 5b where the response from A to B happen precisely when the time fires. Figures 6a and 6b show the state diagram equivalent for their semantics respectively. *Note: The term precisely in discrete time domains means anytime before the clock changes its current value.*



A timer is also used to mark events that happen periodically. Figure 7a illustrates the notation for this utilization based on current trends for interaction diagrams in AUML. Figure 7b illustrates the semantics for a state machine.



In figure 7a, the small sand-clock at the top right corner of the loop notation indicates that this loop is time controlled. Implicit in the semantics of the notation above is the constraint that the system is able to perform the body of the loop before the next loop interaction.

3.3.3 Time-out

A time-out constraint indicates an alternative path in case a time deadline is reached. Therefore the most effective for a time-out constraint is based on the notation for alternative paths in AUML with an indication that the alternative is time sensitive. Figure 8a shows the sequence diagram for a time-based action (such as eBay). The state diagram equivalent is shown in figure 8b.

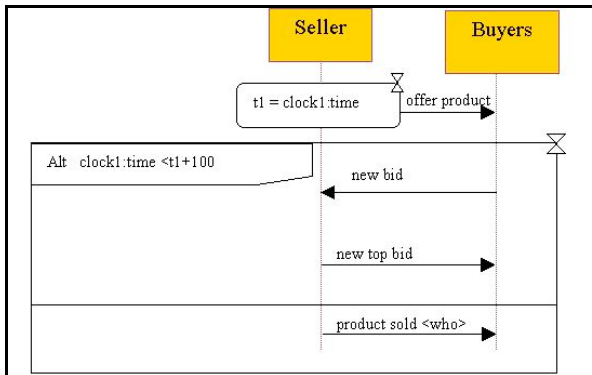


Figure 8a - Time-out constraint representation in sequence diagrams

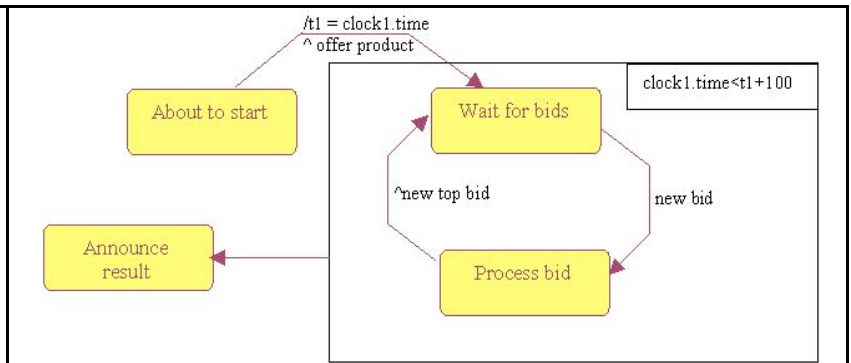


Figure 8b – Time-out state diagram representation

<Challenge:

How can we represent a time-out to break out of time controlled loop IF the body of the loop is not finished on time? >

3.3.4 Composed Constraints

In order to illustrate a compose constraint we will model a system in which two agents with independent time domains respond to a query from a third agent. The order in which the agents respond are not known but the delay constraint allows for a limited period of time after the first one received the message. Observe that for this environment messages are not received instantaneously but the timestamp of the received message is returned (much like an email confirmation). Note: Agent B has visibility the time domains of A and C

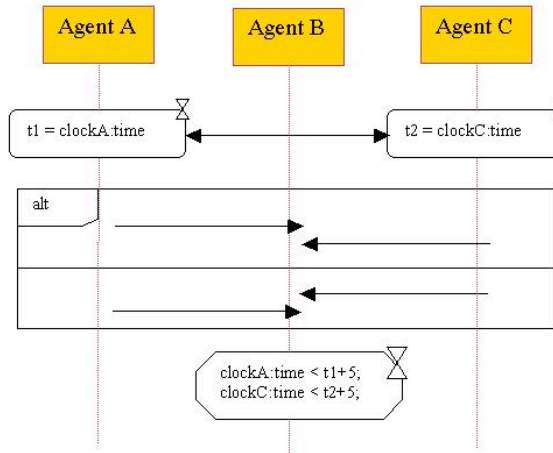


Figure 9 - Composed Constraint

< I don't like this example much, can anyone come up with a better one>

4 Example

< I will introduce examples after the first or second round of discussions>

5 References

[Marely02] Marely, R., Harel, D., and Kugler, H., "Multiple Instances and Symbolic Variables in Executable Sequence Charts", OOPSA 2002, November 4-8, 2002, Seattle, WA

[DAMM01] Damm, W. and Harel D., "LSCs: Breathing Life into Message Sequence Charts", Formal Methods in System Design, 18, pages 45-80, Kluwer Academic Publishers, 2001

[Harel02] Harel, D. and Marelly, R., "Playing with Time: On the Specification and Execution of Time-Enriched LSCs", 10th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'02), October, Fort Worth, Texas, 2002

<and AUML FIPA documents not yet published>