Postponing commitment to preserve opportunities when dynamically assigning new goals to UAVs

Pierre-Yves Dumas

Postponing commitment to preserve opportunities when dynamically assigning new goals to UAVs

Pierre-Yves Dumas
THALES Aerospace Division - 2 Avenue Gay Lussac, 78852 Elancourt, FRANCE
patrick.taillibert@fr.thalesgroup.com

This paper presents our distributed approach to dynamically assign new goals to a fleet of Unmanned Aerial Vehicles (UAVs). Decisions to assign new goals are not instantaneous and they tend to become obsolete because UAVs move continuously. We use the date of commitment to handle this.
This paper introduces how we use a function relating the arrival date to the date of commitment in order to postpone commitment to preserve opportunities. The data of this function are stored in a structure we name commitment vector.

1. Introduction

Continuously moving agents like Unmanned Aerial Vehicles (UAVs) tend to be used in dynamic situations where more and more units are deployed and more and more parameters are taken into account. Decisions to dynamically update the schedules of these agents cannot be considered instantaneous, mainly because human decision must remain central in the decision process. Since agents move continuously, not instantaneous decisions tend to become obsolete. Architectures must be specifically designed in order to handle this. A previous paper of our laboratory [1] presented our distributed approach combining multi-agent and trajectory planning techniques to coordinate several UAVs involved in temporally constrained missions. Our approach uses the *date of commitment* to take decision duration into account.

We consider a decision to be a choice among mutually exclusive scenarios computed out of a simulation. In a dynamic context a scenario tends to stop being an option when time runs. Time runs even while scenarios are being computed and while human are evaluating them. When eventually an order is committed as a result of this evaluation, at the date of commitment, the context must be the one which was the reference when computing the lately evaluated scenarios. If well anticipated, this date of commitment is efficient to make sure that decisions will not be obsolete once made.

Despite being efficient, the raw date of commitment is not easy to use because it assumes that the decision duration can be predicted. Experiences we performed with demonstrators taught us that opportunities are lost if this date of commitment is predicted too soon or too late. We should then predict it as accurately as possible. This is however difficult because the decision duration depends on what scenarios are evaluated and these very scenarios depend on the date of commitment. Therefore there is a dependence loop that leaves us with no other option but an approximate guess based on experience.

To overcome this problem, rather than considering a single date of commitment, we studied $\Delta(\tau)$ the function relating the arrival date to the date of commitment. We discovered that $\Delta(\tau)$ is a non- decreasing function and often displays constant parts we refer to as *steps*. We also discovered that we can use these steps to postpone commitment to preserve opportunities in our dynamic system. This spares complex problems induced by (de/re)-commitment. Moreover the non-decreasing property facilitates the approximation of $\Delta(\tau)$. We store the steps in $\Delta(\tau)$ as a vector of pairs of points of $\Delta(\tau)$. We refer to this vector as the *commitment vector*.

The outline of this paper is as follows. Section 2 describes the system architecture on which our coordination model relies. Section 3 exposes the coordination model we propose. Section 4 presents the commitment vector. Section 5 explains how to use this vector to preserve opportunities in a dynamic system. Then we finally conclude.

2. System Architecture Design

It is now admitted that Distributed Systems of Agents, also called Multi-Agent Systems (MAS) are well suited to design large scale distributed systems made of multiple autonomous and heterogeneous entities engaged in one or more activities, and where coordination is a major issue. Indeed, the different properties i.e. *modularity*, *flexibility*, *robustness*, *scalability* and *decentralization*, required by such systems, are inherent characteristics of MAS. That is why we have based our work concerning multiple UAVs coordination on the multi-agent paradigm.

In the architecture we present, two types of intelligent entities interact: the first one is *human* (Operator) and the second is *computational* (UAVs and Ground Control Station - GCS), as depicted in Fig.1.

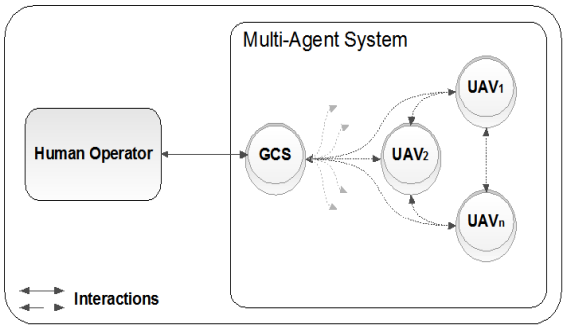

Fig.1. System Architecture Overview

The human operator represents a chain of command. His purpose is to enact and monitor assignment of missions to UAVs. Decision and control continuously remain his in our approach

Computational entities are *intelligent agents* [2]. Basically, an intelligent agent is an autonomous entity able to perceive its environment, to exhibit a goal-directed behaviour and to interact with other computational or human agents.

The GCS agent is an interface between the human operator and the UAVs. It transmits orders from the human operator to the UAVs (such as a new temporarily constrained mission) in one way. It transmits data from UAVs to the human operator in the other way. The GCS can assist the human operator as we will see in section 5.

The behaviour of each UAV agent is driven by a single goal: to perform the different missions requested by the human operator. A mission is to visit a site within a window time. To accomplish this goal, a UAV agent is structured as depicted in Fig.2.

Each UAV agent is made up of four modules. The environment model (module #1) contains data, such as wind field, about the dynamic environment. These data are initially predicted, then they are updated in real-time during the flight by on-board sensors or data emitted by the GSC Agent (module #2). Using the environment model, a UAV agent is able to produce time-optimal trajectories via an on-board planner (module #3) we developed [3,4]. A UAV can dynamically re-plan in-flight its trajectory with its embedded planner. This decentralizes computations and reduces data transfers between the UAVs and the GCS. A UAV agent can use its on-board planner to produce a *hypothetical trajectory* while following its current trajectory. Eventually the 4th module (module #4) executes inner and outer decisions, controls this execution and decides autonomously according the feedback of this control.

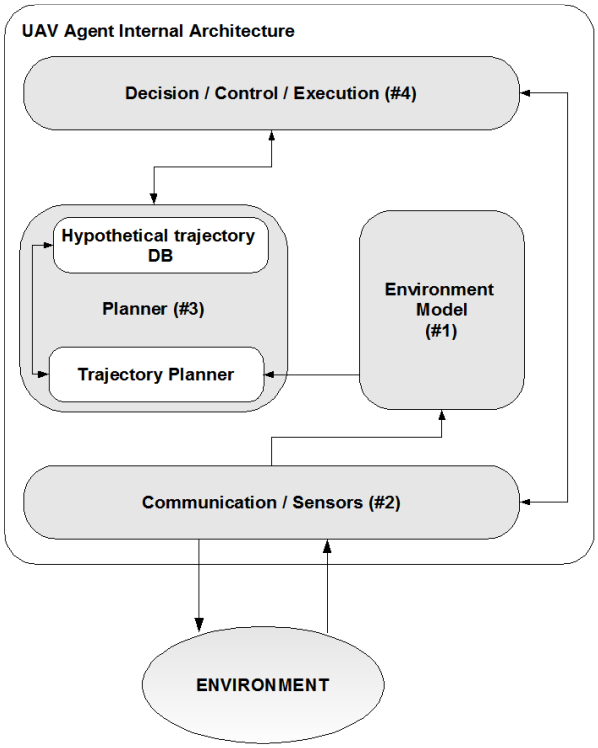

Fig.2. UAV Agent – Internal Architecture

The on-board planner is assumed to give a good answer but not necessarily the best because the speed matters more than the optimality. Other planners than the one we developed could be considered. We do not assume that the same calculation must give twice the same result or that adding constraints should never allow a better result.

Assuming each UAV evolves at a constant altitude, the environment is modelled by a 2-D Euclidean space *P*.
As shown in Fig.3., *P* contains the following items: *sites* to visit (white dots), static obstacles (dark regions) and moving obstacles (dotted circles) and a wind field (grey arrows).

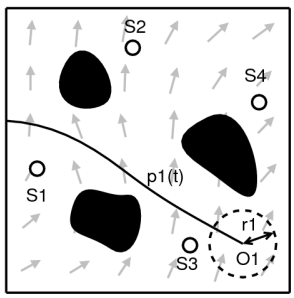

Fig.3. Environment of a UAV Agent

Sites, denoted by S_j ($j \in [1, m]$) are strategic places to be visited by the UAV within a time window $T_j = \left[t_j^-, t_j^+\right]$ to fulfil the *mission* M_j. By convention S_1 and S_m denotes the take-off site and the landing site.
Static obstacles are closed surfaces of any shape, corresponding to dangerous or inaccessible areas.

Moving obstacles O_l are disks of radius r_l. They correspond to punctual mobiles surrounded by a circular safety zone. The position of O_l is given at every time t, and denoted $p(O_l,t)$.

Finally, the *wind field* W is known by measurement or forecasting. Data about W are thus discontinuous, defined on the nodes of a mesh (grey arrows).

3. Coordination Model Overview

The basic scenario we will refer to is a fleet of several UAVs controlled by a human operator through a GCS. While each UAV is completing its own list of missions initially planned, a new mission (a new site to be visited in a time window) is requested. Every mission is to be completed by a single UAV.

3.1 Goal (new mission) assignment problem

The calculation of the trajectory, as we explained in section 2, is handled in a distributed manner by the UAVs. Therefore a convenient way to solve the goal assignment problem is to use the well-known negotiation-based interaction protocol called *Contract Net Protocol* (CNP) [5]. This protocol relies on the human contracting mechanism (announce, bid, award cycles) and allows tasks to be distributed among a group of agents.

Whereas in the original CNP the roles of agents are not necessarily specified in advance, they are fixed in our approach. The human operator plays the role of the *manager* (announces, receives, evaluates bids and awards to a contractor). The GSC assists him. The UAVs play the role of potential *contractors* (receives, evaluates, bids or declines, and performs the task if awarded). The bids emitted by the UAVs during the goal assignment task are a set of data returned by their on-board trajectory planners.

The manager should decline each bid he is not interested in as fast as possible, in order for the matching UAV to be released of every constraints related to the tender. This is important if several tenders can happen to be concurrent, possibly with several managers. Such a situation is the topic of another of our on-going works.

The whole process cannot be considered instantaneous, especially since the human has to evaluate bids. In the meantime, UAVs continue flying and a bid may not be anymore performed when eventually awarded. To overcome this problem we use the date of commitment.

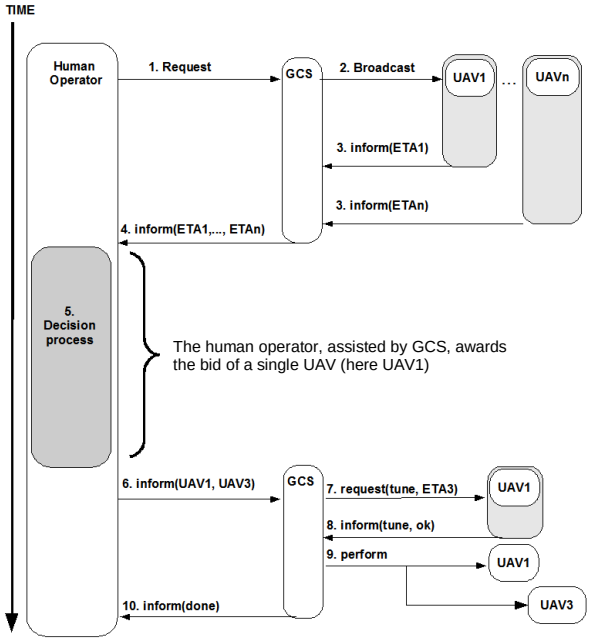

Fig.4. Coordination model

3.2 Date of Commitment

As we mentioned before, trajectory computation and goal assignment take time. In order to provide a coordination model taking into account duration of both computed planning and human deliberation, we proposed [1] to introduce a mechanism commonly used in the human coordination: a *date of commitment* (denoted by τ). This date of commitment is a date ahead used instead of the current date to compute data to drive a decision in order for this decision to be up to date when its resulting orders are committed. In our assignment model this happens when the human operator, assisted by the GCS, eventually awards a single bid.

Despite being efficient, the raw date of commitment is not easy to use because it assumes that the decision duration can be predicted. If this duration is not accurately predicated, the date of commitment will be more constraining than it should and opportunities will be lost. In our assignment model the duration of the awarding process, performed by the human operator, assisted by the CGS, depends on which bids are received, what is unknown when the tender starts and when the date of commitment has to be chosen.

To overcome this problem we studied what happen when different dates of commitment are chosen. That led us to use $\Delta(\tau)$ the function relating the arrival date to the date of commitment whom we store the main data in the commitment vector instead of a raw date of commitment.

4 Δ(τ) and the commitment vector

Our global concern is to make UAVs perform missions when minimizing the flight time. In the fallowing example the speed is constant, so spatially longer means also temporarily longer. We consider also a single UAV to simplify notation. We eventually assume that our planner returns always the best trajectory (whose the duration is the shortest) to schedule missions of an UAV. Reality is generally much more complex but the purpose of our example is to focus on some properties of Δ(τ).

4.1 Presentation

Before assigning a new mission, our UAV is flying following its own initial trajectory T_0. This initial trajectory T_0 is expressed as a succession of n waypoints denoted by w_i ($i \in [1, n]$). The first waypoint w_1 matches S_1, the take-off site (grey with number 1), the last waypoint w_n matches the landing site S_m (grey with number m), some other waypoints are related to a site (here a single one, grey with S_2). Every waypoint w_i is met at a meeting time τ_i.

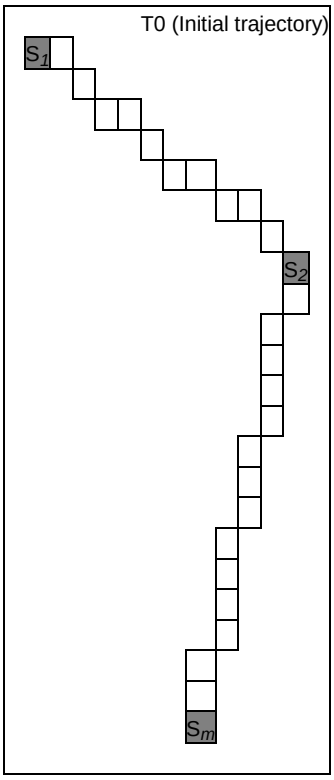

Fig.5.1. Initial trajectory T_0

Let us now consider a new site (dark grey with S_x) we want the UAV to visit to perform a new mission. In our example we do not consider the constraint of the time window.

For every date of commitment τ_i there is a matching trajectory $T\tau_i$ that allows to visit S_x in addition of S_1 to S_m. $T\tau_i$ is the same that T_0 until τ_j. After τ_j, $T\tau_j$ can (and will likely) be different from T_0 in order to visit the new waypoint. On the next column there are four samples of such trajectories. The new trajectory is grey and on the top when the initial one is white and partly hidden under the new one.

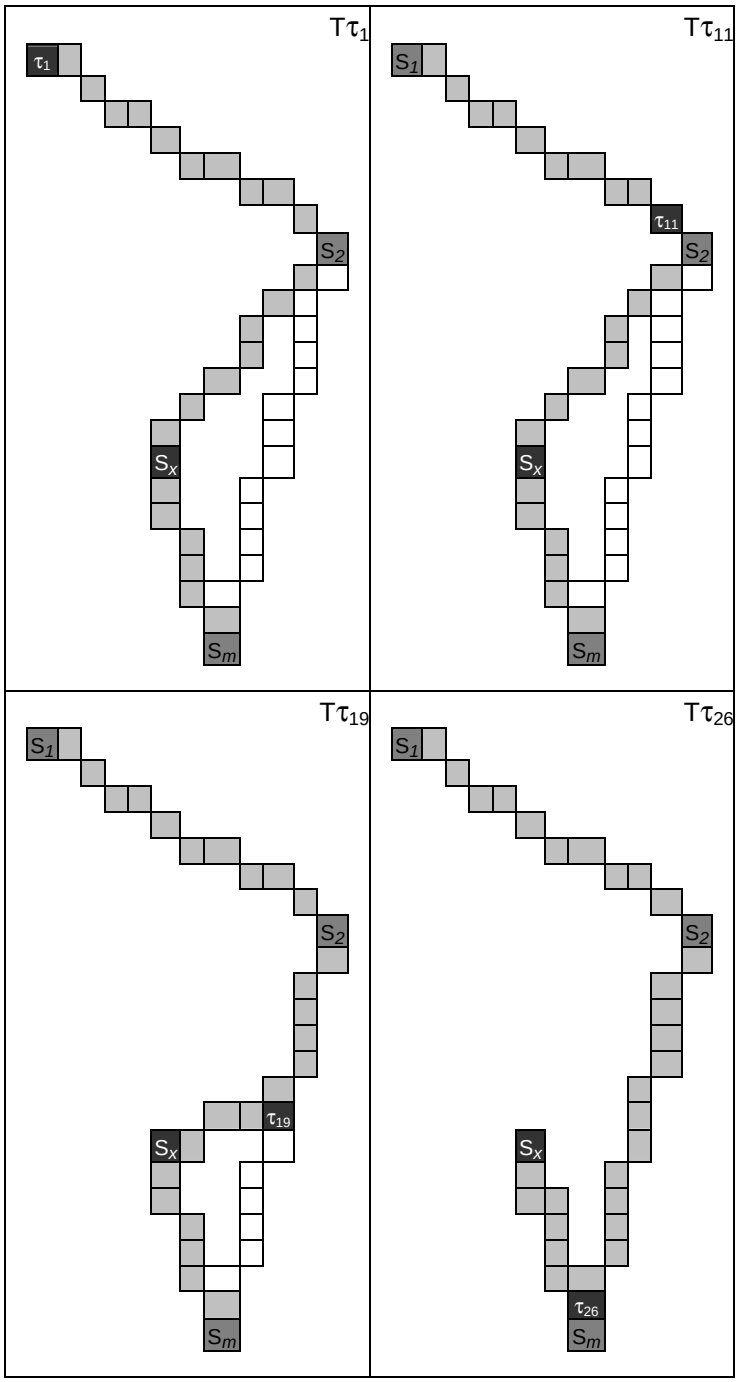

Fig.5.2. Four trajectories as sample $T\tau_i$

We can see in our example that every $T\tau_i$ is longer than T_0. The reason is simple: adding a new waypoint we add new constraints. Since our planner minimize the flight duration, the trajectory length of $T\tau_i$ is necessarily longer than the one of T_0, and therefore less good.

Also the later is τ_i, the more the UAV must wait before changing its trajectory τ_i relatively to T_0. This is obviously a constraint that becomes stronger when τ_i becomes later. Therefore the bigger i, the later τ_i, the longer $T\tau_i$.

We define $\Delta(\tau)$ the function relating the arrival date to the date of the commitment to visit S_x in addition of S_1 to S_m. Below is the curve of this function in our example.

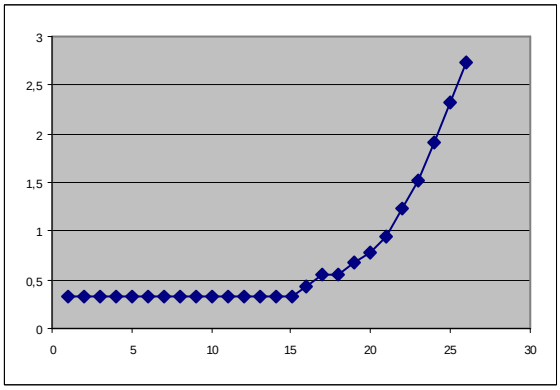

Fig.6.1. Example of $\Delta(\tau)$

We can see on our example that there is a step until half way, a bit later after visiting S_2. There is also a small step a bit later.

We consider that the second step is too short to be meaningful, it may indeed be the mere result of how the planner works. More generally we consider that a step is made of at least 3 consecutive points with an equal arrival date. We can see in our example that there is only one such a step, all over the first half of the curve.

Such steps in $\Delta(\tau)$ are very common. The counter-part situation of those steps is when every postponing of the date of commitment means giving up on a solution that was the best until then. Especially when getting close to the landing site, the new trajectory looks more and more like the initial trajectory plus a round trip to the new point, which is worse and worse.

Since we are interested in steps, we store the points matching their bounds in a set of data we name commitment vector. We also store in this vector the points matching S_1 to S_m. We eventually store the trajectories matching these points in order to be able to use them if we want so, without a recalculation that may in deed give another result depending on the inner process of the planner.

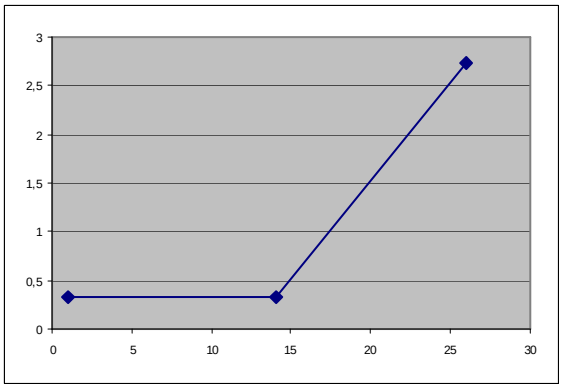

Fig.6.2. Example of commitment vector

4.2 Calculation

It would be expensive to build $\Delta(\tau)$ if we call the planner once per τ_i. However since this function is not decreasing and display steps, we can pick points until we get a result with the desired accuracy. Especially if we get two points with a same delay, we can spare calls to the planner for every τ_i between these points.

This function $\Delta(\tau)$ has other characteristics. For example steps often end just after visiting a site in a way similar of what typically happen before landing. Such characteristics can lead to find out heuristics.

Since we introduced $\Delta(\tau)$ and the matching commitment vector and how to calculate it, let us see what it is useful for.

5 Using steps to postpone commitment

The function $\Delta(\tau)$ makes possible for the manager to see how late will be a UAV to perform a new goal depending on the date of commitment. Every step of the commitment vector means that for the matching period, commitment can be postponed without delaying the landing of the UAV. We can use this possibility in our dynamic context where new goals appear.

In the CNP we saw sooner in the third section, the default behaviour would be to award the UAV with the smallest delay caused by the new mission and to immediately commit the matching new schedule. Let us imagine we have this opportunity when the current date is included in a step. Since we are in a dynamic context where new goals can be added at anytime, a second new goal may appear before the end of this step. It may happen also that the schedule committed with the first new goal leaves no opportunity or only bad opportunities to perform the new mission while alternative and better organisation would be possible now that we know these both missions. Such alternative and better organisation, if exists, may require complex operations such as (de/re)-commitment [6]. It would have been more efficient to wait to commit, since the step revealed by the vector of commitment was saying we could do it without delaying the landing.

This way, we can use steps in the commitment vector to postpone commitment until the end of the current step, if exists, in order to preserve opportunities for incoming new missions which we do not know yet anything about. Since steps are common, according the many experiments we performed, the system becomes much more reactive.

To take advantage of this situation, the human operator, as a manager of our CNP, is not requested to select a single UAV but as many as he wants. Such selected UAVs are not automatically awarded. The CGS, as an assistant of the manager, is the entity that will eventually award a single UAV among them. It should be noted once again that in our approach final decision remains to the human operator since even if he is not the last one to choose, he is the one who choose the domain of the final choice.

Conclusion

We presented an approach to dynamically update the schedule of a fleet of mobile agents, for instance Unmanned Aerial Vehicles (UAVs). This approach is based on a Contract Net Protocol (CNP) where the CGS, assisting the human manager, postpones the commitment according the commitment vector, based on $\Delta(\tau)$, the function representing the delay at the new arrival waypoint to visit, in order to be efficiently reactive. The decision remains to the human.

The strongest characteristic of $\Delta(\tau)$ is to be not decreasing, and to often display steps. This characteristic is useful for both calculation and utilisation of this function. Other characteristics exist.

We expect $\Delta(\tau)$ and the commitment vector to hep us to solve complex problems such as (de/re)-commitment.

References

1 Soulignac M. et al. *Combining Multi-Agent Systems and Trajectory Planning Techniques for UAV Rendezvous Problems*. COGIS 2007.

2 Wooldridge, M.J. Intelligent Agents. In Gerhard Weiss (Ed.). Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence. The MIT Press, MA.

3 Soulignac, M et al. *Fast Trajectory Planning for Multiple Site Surveillance through Moving Obstacles and Wind.* Proc. Of the 25th Workshop of the UK Planning and Scheduling Special Interest Group (planSIG), 2006, pp 25-31.

4 Soulignac, M et al. *Multiple Path Planning using Wavefront Collision.* Proc. Of the IEEE/RSJ International Conference on Intelligent Robots and Systems (2007 IROS), *to appear*.

5 Smith, R.G. *The Contract-Net Protocol: High Level Communication and Control in a Distributed Problem Solving*. IEEE Transactions on Computers, 1981, pp 1104-1113.

6 Jiri Vokrinek, Antonin Komenda, Michal Pechoucek. *Decommitting in Multi-agent Execution in Non-deterministic Environment: Experimental Approach.* AAMAS 2009.