

Integration of Multiple Planning Technologies for Power Management

D.A. Findlay, M.K. Hook, D.W. Long and R.T. Watkin
Roke Manor Research Ltd, Romsey, Hampshire, SO51 0ZN

Abstract

This paper reports on work undertaken within the SEAS DTC Innovation Fund project IF059 'Integration of Multiple Planning Technologies for Power Management' [1].

The SEAS DTC PPEM theme is researching an Intelligent Power Management capability which can mediate between the requirements of uninhabited vehicle subsystems and to operate them in accordance with the requirements of a mission plan. It is desirable, though not an essential requirement, that such a capability can manage power generation systems which cannot support all their uses simultaneously. It is suggested that this can be arranged by employing a power management element with defined interfaces to mission planning and execution control and to the subsystems themselves. Whilst it is possible to create a single monolithic Intelligent Power Management system which understands the power needs of all of the uninhabited vehicle's ancillary systems and which can generate a power-usage plan which will sequence their use to achieve mission goals; it may be advantageous in the long run to separate resource-constrained operation sequencing in an autonomous system from overall on-board mission-task planning; that is, to employ a hierarchy of planners on board the vehicle. This paper reports on the architectural implications of this approach.

Keywords: Power management, multiple planners, planner architectures

Introduction

Existing unmanned vehicle systems are, largely, tele-operated systems and as such, it is eventually up to human decision making how the various power systems of the vehicle interact. In order to create flexible solutions, the existing approach has been to ensure that as far as possible, the vehicle subsystems can be operated independently and simultaneously, without particular regard for limitations of the power generation chain. This approach inexorably results in over-provision of power generation or storage capacity in order to allow for future technology insertions.

The approach being addressed by the PPEM theme is to incorporate intelligent power management that can mediate between the requirements of UxV subsystems and to operate them in accordance with the requirements of a

mission plan, but without requiring that the power generation and storage systems can support all their uses simultaneously.

This paper examines the implications of a hierarchical multiple-planner approach. At one level, a mission-task planner will receive (possibly human-generated) mission goals and will generate a series of achievable sub-goals expressed in terms of movement and the operation of sensor or effector suites. The individual sequencing of operation of individual motors, sensors or effectors must be planned so as to avoid operating all simultaneously, yet at the same time the platform must achieve its mission goals. Inevitably there will arise conflicts in which individual subsystem sequencers will seek to have power made available at the same time as others. Successful management and resolution of such conflicts will be critical to the successful implementation of this approach.

Motivation

The concept of power management for autonomous vehicles by taking a subsystem view is proposed in ADD020 [2]. The question arises of quite which subsystems are being discussed, and how they might be composed. It is apparent that the physical assets onboard an autonomous vehicle may be categorised in a number of ways, including: directly from the assets; by asset function; by criticality, i.e. those assets essential for vehicle operation and survivability regardless of mission, and those other subsystems containing non-critical but task-essential assets; and as power sources, stores, and sinks.

The motivations for intelligent power management include an anticipated increase in system complexity, and a desire for reduced over-provisioning of autonomous vehicles. Solutions which address these motivations will clearly have an impact on platform design, affecting both the hardware on the platform and the software managing that platform. Intuitively, as systems become more complex and over-provisioning is reduced, platform software must become more intelligent – in some sense ‘doing more with less’. Distributed architectures are envisaged to play a growing part in the design of autonomous platforms.

Levels of Planning and Control for Power Management

In this section we present a taxonomy of power management systems with different levels of autonomy. This analysis suggests that the difficulty of providing management even at the level of admission control (the ‘level 1’ system defined below) is considerably greater than in existing systems which use human oversight. This level of autonomy already suggests a set of strong candidate interface elements, in order to allow the mission planning system flexibly to specify subsystem behaviour, and also to allow the power management

system to gather enough information to perform admission control.

Level 0 – A level 0 power management system takes the approach that is commonly adopted in systems today. That is, ensuring that power is available to meet subsystems’ demands is achieved by *over-provision*. In this approach, there is really no dynamic planning or control; essentially all planning decisions were made during the system design process.

Level 1 – A level 1 power management system implements a process that is referred to as simple ‘admission control’, by analogy with the similar process that is performed in mobile telecommunications networks. This approach implements *control*, but still no significant *planning*. Operation of a level 1 power management system can be quite simple until a request is made specifying a power profile that cannot be met within the constraints of the available power and what has already been allocated to other devices. At this point, there are fundamentally three options: to deny the request; to reallocate power that had previously been allocated to other devices; and to increase the amount of power that is available so that the new request can be met.

Level 2 – A level 2 power management system augments the operation of a level 1 system by adding an element of *reactive planning*. This planning is designed to meet two goals: to coordinate the planning of the various subsystems so that power demand conflicts are resolved during the planning phase (rather than, as in the level 1 approach, during the plan execution phase); and to plan the use of power generation facilities in order to meet the (planned) power demands of the various subsystems. As with level 1, options exist for handling a request made specifying a power profile that cannot be met within the constraints of the available power. In addition to the level 1 options, level 2 has a further option at this

point: it could deduce a set of devices whose power demands are in conflict, and mediate negotiation between those devices' subsystems' planners in order that they should arrive at a mutually agreeable plan that resolves the power demand conflict.

Level 3 – A level 3 power management system augments the operation of a level 2 system by adding an element of *proactive planning*. This enables a subsystem to devolve detailed planning for the use of a device to achieve a task to the power management system. Of course, for this to be possible requires that the power management systems have a deeper understanding of a device's power demands than is given by the simple power profiles required by level 1 and 2 approaches. The system must also be aware of even more resource constraints. The difficulty in handling these constraints is only compounded if the constraints are not constant.

Planner Architectures

In this section we present an overview of factors which influence the architecture of planners and plans. Different power management requirements may be best addressed by different architectures, and incorporating everything presented here might be overkill for all but the most complex systems.

Figure 1 presents a taxonomy of planner architectures – ways of arranging the planner functions in an autonomous (vehicle) system. We regard this taxonomy as presenting a variety of architectures suitable for implementation for own-platform goal and power management planning in an autonomous system consisting of one vehicle.

The taxonomy sets out a view of planner architectures. Clearly, the diagram is asymmetric since a single centralised planner cannot, by definition, be decomposed further. On the other hand,

once planning is distributed in the system, a number of ways are shown to achieve this.

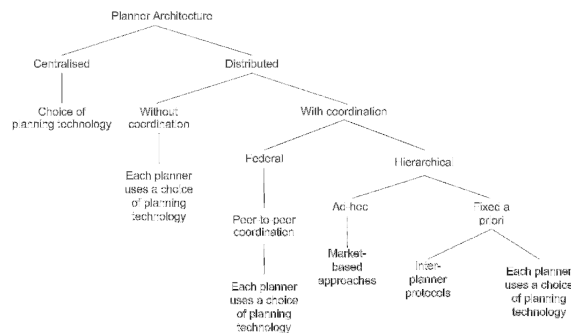


Figure 1: A taxonomy of planner architectures

Given the taxonomy of Figure 1, the choice of architecture is largely driven by an analysis of the requirements of a particular, given, system. For these reasons, this section does not make explicit reference to known systems. Furthermore, we do not impose a particular internal structure in any one node in an architecture of planners; this is a level of detail beyond the scope of this paper.

The discussion above of power management subsystems suggests the possibility of a collection of subsystems, each with its own planner. For example, there may be independent planner entities for each one of the communications, sensor suite, and power generation subsystems. In addition to the subsystems themselves, it is reasonable to consider the existence of an overarching 'coordinator' planner function above the subsystem planners. Equally, the subsystems might have a non-trivial internal structure. Hence there are several possible ways of distributing the planning function (and planning entities) in a system. One distinguishing feature of such arrangements is whether they are *hierarchical* or *non-hierarchical*.

Hierarchical architectures are a natural fit to the subsystem view of an autonomous system. The principles of subsidiarity and localised decision-making, under high-level supervision and coordination, are appropriate for combined goal and power management planning in an autonomous

vehicle demonstrator. The planner hierarchy may implement intelligent power management at the levels 0-3 identified in above, appropriate to the application.

Interfaces between Planners

When multiple interacting planners exist in a system, a number of interfaces can be envisaged. The interfaces may be defined, deliberate and fixed, or may be application specific ad-hoc interfaces, or may be something in between. Factors such as interoperability and clarity of design motivate the desire for a uniform defined interface between planners. Notionally, planners implement subsets of some generalised interface and provide null implementations of the complement, depending on individual planner sophistication.

For any interaction to take place, communication channels must be established, and message syntax and semantics must be defined. To some extent these characteristics are domain-dependent; nevertheless some work has been done in other SEAS DTC projects into these issues (IF052 [6], IF058 [7]).

A number of questions need to be addressed regarding planner interfaces: (1) where are the interfaces, (2) what are the communicative acts, and (3) what information needs to be passed around. We address these by firstly identifying the interface classes, then by enumerating the information classes, and finally by merging the two to map information to interfaces.

The four principal planner interface classes shown in Figure 2 are to mission control, to other planners, to situational awareness, and to plan execution. Note that not all of these interfaces need to exist in every system. For example it is possible to construct a hierarchy of planners in which peer planners (those at the same hierarchical level) do not interact. Conversely it is possible to construct a

planner architecture comprising entirely of peers, such as a contract net, in which there are no ‘superior’ or ‘subsidiary’ planners.

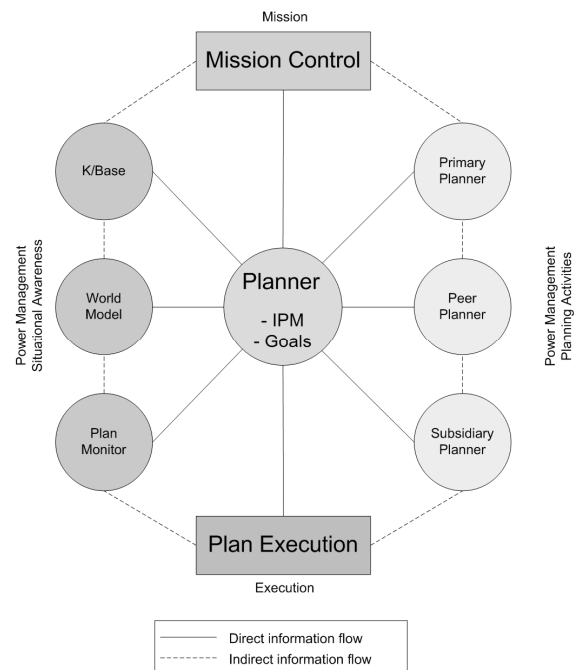


Figure 2: Power management planner interfaces

The transactions between entities in a multi-planner architecture such as in Figure 2 exchange the following classes of information:

- Goals – may be imposed from outside the system (from Mission Control), or delegated within the system (from superior planners)
- Constraints – physical, temporal, resource, situational, to include constraints derived from power management considerations in the system (again, from Mission Control or from superior planners)
- Task allocation protocols (between planners)
- Plan (solution) structures (between planners, and for execution)
- Plan monitoring information (from the situational awareness picture)
- Situational Awareness (SA) picture
- Protocol-specific success / failure / data transactions

A system of planners can be seen as a collection of independent but interacting individuals. The multi-agent paradigm of distributed reasoning entities is an appropriate analogy [8].

Information Propagation

When several planners cooperate to achieve power management within a vehicle, information which describes the planning problem must be exchanged between them. The basic classes of information are goals, constraints, agendas and conflicts. These are illustrated in Figure 3, and are now described in turn.

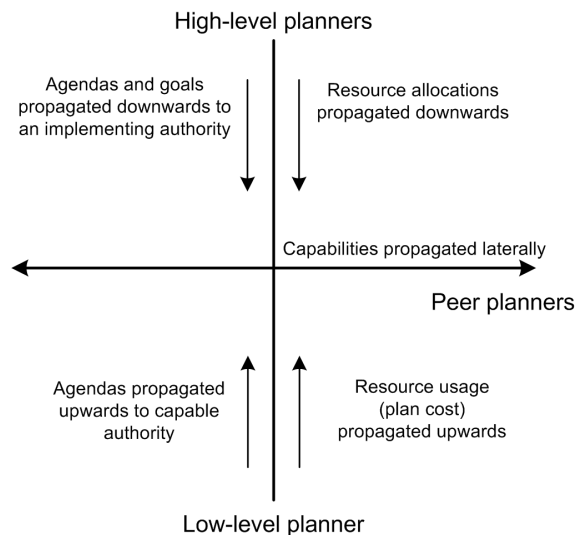


Figure 3: Information propagation

Goal Propagation

One method of goal propagation is to pass the goal in its entirety from planner to planner, in the hope of finding a planner which can provide a complete plan solution. This method is undesirable for a number of reasons. Firstly, if the existence of the capable planner were known outside the system, the goal should be assigned to it directly. Secondly, the degree of specialism in any one planner renders the system inflexible to modifications to goals.

An alternative method is to decompose goals into subgoals; in other words use a divide-and-conquer approach. Following decomposition, subgoals are propagated

through the system. This approach has advantages, but also significant technical challenges.

The first advantage is that planners can be developed which provide plan solution 'building blocks', for example waypoint sequence generators or task schedulers; combinations of such components are notionally used to construct whole solutions. Secondly, new planners with their own building block specialisms can be introduced into the system, augmenting system capability. This contrasts well with the equivalent in the 'whole goal' scheme, in which a fully capable planner would be required to handle small deviations from the existing repertoire. Finally, part-solutions can be formulated concurrently by independent planners, thus reducing planning time.

It is difficult to make useful generalisations about goal decomposition because each goal has (possibly latent) contextual semantics. Ideally, the actual goal decomposition matches the capabilities of other planners in a particular system, and thus only decomposes a goal into achievable subgoals. In general this is not a reliable assumption, but the contextual semantics of any one goal might be of help in this regard. Note that however goals are to be decomposed, it is assumed that the system design incorporates planners capable of solving goals at some level of decomposition. The processes described here may have to be applied recursively to reach these levels.

Having decomposed the goal, the next step is to propagate subgoals amongst the planners in the system. This can be achieved in a number of ways. A crude and inefficient approach is to propagate every subgoal to every planner in the system. Market-based solutions such as contract nets provide a more elegant approach, but one that still requires the task announcements to be global. An alternative

to global task dissemination is for planners to advertise their capabilities, for example with a service registrar which is consulted for matching tasks with planners (a yellow-pages approach).

Constraint Propagation

Just as goals are decomposed for propagation in the system, an equivalent process can be applied to constraints. This process is one of constraint refinement. It is possible that a high-level constraint specification is not detailed enough for an effective power management schedule to be deduced. Therefore the constraints must be refined until such a schedule can be set out.

Constraint refinement is essentially an optimisation problem, which is guided by a number of principles. Firstly, solutions within a constraint set have a goodness measure, which is application-dependent and typically heuristic. Secondly, constraint refinement is an iterative process which starts with an initial feasible solution solving the goals within the bounds of the initial constraint set. Finally, note that the globally-optimal solution for a particular problem will not in general be found; if it is found then this is by chance rather than design.

Having formulated a solution which satisfies the (vague) initial constraints, it may be desirable to explore 'nearby' solutions, in other words those in which some of the parameters of the initial solution are perturbed. However, this is not a reliable approach, because the way in which a solution depends on its parameters is not known. For example, if a solution requires two vehicles to be in the same area at a known time of day, then arriving too early or too late might break the solution; therefore the plan is intolerant of perturbations to arrival time. Mathematically speaking, nearby solution searching is unreliable because the geometry of the multi-dimensional solution space is not known: the solution space

might not be smooth in some dimensions, or even have discontinuities.

A possible scheme for constraint refinement is as follows. Initially, the vaguest possible constraints are passed to the planners. An initial feasible solution is composed by the planners which contains descriptions of the *flexibility* of the constituent solutions. Flexibility in this sense means the tolerance of the plan to perturbations to its initial conditions; examples include whether an operation in the plan has to happen at a particular point in the plan; and how resource consumption is distributed in the plan. The initial solution and statements of flexibility are then treated together as an arrangement problem, in which solutions are arranged according to their stated flexibility. The arrangement then permits tighter constraints to be derived.

Agenda Propagation

In the context of this paper, an agenda is a statement of processes or behaviours which an entity in the system desires to see enacted. In general, agendas are weaker than goals: a planner will seek to satisfy its goals even if items on its agenda are not enacted in the resulting plan. However, agenda items held by a high-level planner might manifest themselves in the goals imposed on lower-level planners, should the higher-level entity have the freedom to enact its agenda. Note that the entity holding an agenda need not necessarily enact that agenda (perhaps because it cannot, by itself): thus agendas may need to be propagated through the system if they are to be enacted.

Agendas can be incorporated as part of a planning problem by elevating agendas to the status of goals, provided that the agenda is propagated to those planners in the system which manage goal decomposition and distribution. However, if several agendas exist for power management, then scheduling and merging agendas becomes

difficult. For example, several competing agendas might exist for making use of a rechargeable power storage unit, and it may be that these agendas cannot all be enacted simultaneously.

Solution Recombination

The previous discussion describes how goals set from outside the system can be decomposed into subgoals, and how those subgoals can be distributed to planners within the system. This section considers how the resulting planned (partial) solutions to subgoals can be recombined to form a whole plan solution, if required.

Typically, recombining plan solutions is subject to some structural constraints. Firstly, there may be a temporal constraint, and solutions need to be recombined such that they appear in the correct order. This is similar to the problem encountered in Hierarchical Task Network planning, in which the recursive method expansions must be recombined to produce a plan in the form of a partially-ordered sequence of actions. The paradigmatic HTN algorithm set out in [9] uses ‘critic’ functions to achieve this; the detailed mechanism of each critic is application-dependent.

Secondly, there may be arrangement constraints resulting from the way in which the goal is decomposed. This is analogous to the familiar technique for handling parentheses in arithmetic expressions: the ‘innermost’ parentheses are evaluated first, and those partial solutions then contribute to some other part of the problem.

Resource Conflict Resolution

When goals are distributed by a primary planner, plan solutions are returned for recombination. Because plan solutions are formulated independently, resource conflicts can arise. Strategies for conflict resolution are therefore required.

One discriminating characteristic of conflict resolution strategies is whether resolution is performed by some central controller, or in a distributed fashion. A centralised resolution agent acts on the fullest available information relating to all conflicts in the system and, notionally at least, can resolve all conflicts. A difficulty of this approach is that large numbers of intricate conflicts could give rise to computationally expensive problems. This difficulty is mitigated to some degree by a distributed approach, in which individuals negotiate only with those other individuals with which their plans conflict. Whilst this is conceivably more efficient, there is no guarantee that local negotiations do not introduce new conflicts with previously unaffected individuals.

Plan Execution Monitoring

Several subsystem plans can exist and be executed concurrently in the system. Execution of a plan is separated from planning, and furthermore from the execution of other plans, although at some level, the combined effect of plans must fulfil mission goals.

Plan execution monitoring can be implemented in a number of ways:

1. Periodic comparison of the world model against the plan (or at least the expectations derived from ‘the plan’) at some appropriate frequency, and
2. By notification from the world model of events or conditions.

Plan execution monitoring has a substantial heuristic element. Consider, for example, a UAV following a waypoint sequence; in which case it may be appropriate to monitor the UAV position, speed and heading at the expected arrival time at each waypoint, to compare reality with the plan. Naturally, it is unreasonable to expect the UAV to be at a precisely-specified lat / long / altitude at a precise time, so some

heuristic tolerance needs to be built in and this will differ depending on application. A small, slow-moving UAV might tolerate being within 1m of the expected waypoint at the expected time; a larger fast-moving UAV might require a larger tolerance sphere.

Each subsystem may possess its own local working repository of information – its local blackboard – and the architecture may also contain a ‘global’ world model. The global world model stores information required for subsystem monitoring but which cannot necessarily be sourced directly from within the subsystem itself; a GPS time source is a typical example. The additional input of global information is shown in Figure 4.

Hierarchical architectures offer the possibility of ‘vertical monitoring’, in which monitoring of subsidiary planners forms part of monitoring at a given level. Monitoring of subsidiary planners can occur by the same periodic or notification based schemes as already outlined.

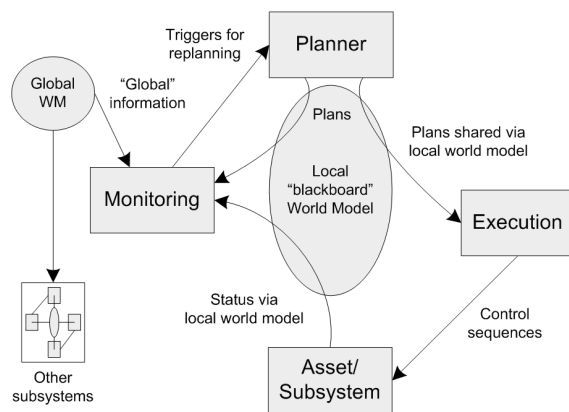


Figure 4: Plan execution monitoring

When plan repair / replanning is triggered, the subsystem planner may be able to resolve the situation and formulate a new or repaired plan itself. To some extent, plan repair / replanning can be contained within subsystems. In other cases, this is clearly not possible, for example on asset degradation or failure, or when insufficient resources are allocated to a subsystem to attain its goals.

A Model for Planning in an Unstructured World

Traditional AI techniques, such as the Stanford Research Institute Problem Solver (STRIPS), tend not to be resilient against a changing world. This section describes a bio-inspired planning strategy (based on Suchman’s concept of situated actions [3]), which is inherently robust in the context of a changing and unstructured world.

To achieve desired purposeful behaviour in an unstructured world, our approach starts with a goal, or more properly an (ordered) set of goals. In addition we require a repertoire of behaviours and the ability to select behaviours based on our current context and desired goal(s). We must also have the ability to monitor the current (changing) context and assess the continuing applicability of the currently executing behaviour.

A goal is seen as a set of predicates to be satisfied. Those predicates might be statements in terms of location and time as well as other attributes of a desired condition. A goal can be recursively decomposed into an ordered set of subgoals, taking into account the initial context.

In this model of planning, a plan is only ever a set of goals (desires, intents). It does not capture how to implement those goals. Early AI planning solutions did not have sufficient computing power available to allow much in the way of run-time goal expansion complexity, thus traditional planners had to expand up tasks to the finest level of detail prior to commencing execution – e.g. as proposed by Vere for detailed spacecraft orientation and sensor sequencing during planetary flypasts by Voyager [4]. Even in the space domain, more recent work allows for rather more in the way of run-time complexity [5].

At run time, the goals are recursively decomposed using a lazy-computation

inspired approach (i.e. we only decompose what we have to for the moment). Note that to avoid getting trapped in far-from-optimal or dead-end states, the goal-rewriter may need to assess the likely complexity of each derived subgoal and, if appropriate, look ahead a few steps, rather like an expert chess player. A look-ahead rewriter notionally removes the need to backtrack. Pragmatically, a system might not physically be able to backtrack in a given situation. There is a trade-off between agility (not looking too far ahead) and computational overhead / paralysis of choice (looking too far ahead in a combinatorially-explosive decision tree).

Behaviours are analogous to Suchman's situated actions [3]. A behaviour is an instantiation of a pre-defined (possibly with goal-directed parameterisation) script or similar 'atomic' action. Behaviours are associated with applicable contexts. Changes to the current context may render the current behaviour inapplicable and necessitate an alternate behaviour.

Basic training is equivalent to developing an (initial) repertoire of behaviours, together with the ability to appropriately instantiate those behaviours based on context and a goal-directed parameter set. Further, the system must be able to recognise the current context and interpretation of the semantics implicit in those facts.

Skilled competence involves being able to efficiently recognise relevant elements in the current context and then to select and instantiate an appropriate behaviour that will help lead towards the desired goal state, based on that current context. Skilled competence further requires the ability to recognise that the prevailing context has now changed significantly and that the current behaviour is no longer applicable, triggering selection of a new behaviour.

We also require the ability to recognise that the currently targeted goal state has either

been attained, or that it is no longer applicable and that an alternate goal must be selected by the goal-rewriter.

The challenge is thus to efficiently recognise critical elements in the current context pertinent to behaviour selection and to the current goal, and to thence select an appropriate behaviour from a set of plausibly applicable behaviours.

Further Work

Hierarchical architectures are a natural fit to the subsystem view of an autonomous system. The principles of subsidiarity and localised decision-making, under high-level supervision and coordination, are appropriate for combined mission and power management planning in an autonomous vehicle demonstrator. Our work sets out options and guidance for logical and physical distribution of the planning function. The planner hierarchy may implement intelligent power management at the levels identified in this paper, appropriate to the application. Therefore we recommend that hierarchical architectures, with the options of distribution and planning sophistication, are considered for inclusion in application architectures.

We note that in order for planners to produce realistic and feasible plans, they must operate on situational awareness which reflects the current world state. Furthermore, when plans are executed, progress against the plan must be evaluated with respect to the evolving situational awareness picture. Monitoring functions are essential counterparts to planning in an autonomous system, and it is recommended that they are incorporated in application architectures.

We have proposed a model of planning in an unstructured world. We recommend a short algorithm design and software development based on this, to produce a statement of the planning algorithm and

associated data inputs, and a proof-of-concept software implementation in a representative planning problem in own-vehicle power management planning.

Systems (SEAS) Defence Technology Centre established by the UK Ministry of Defence.

References

- [1] R. Watkin, *Integration of Multiple Planning Technologies for Power Management*, SEAS DTC IF059 Final Report IF059/roke/001. (February 2009)
- [2] M. Johnson, *Concepts and Options for Power Management on Autonomous Vehicles*, SEAS DTC ADD020 Final Report ADD020-01, Draft. (March 2008)
- [3] L. A. Suchman, *Plans and Situated Actions, The problem of human machine communication*, Cambridge University Press (1987)
- [4] S. A. Vere, *Planning in Time: Windows and Durations for Activities and Goals*, IEEE trans. on Pattern Analysis and Machine intelligence, PAMI-5, No. 3, 246-267 (May 1983)
- [5] A. Barrett, *Domain Compilation for Embedded Real-Time Planning*. In Proceedings of the Fourteenth International Conference on Automated Planning & Scheduling, Whistler, British Columbia, Canada, AAAI Press (June 2004)
- [6] A. Centonza, A. Ford, M. Hook, R. Watkin. *Dynamic Role Assignment for Mesh-Based Autonomous Systems*, SEAS DTC IF052 Final Project Report, IF052/roke/0002, Issue 1. (February 2008)
- [7] R. Watkin. *Semantics for Power Management*, SEAS DTC IF058 Final Project Report, IF058/roke/001, Issue 1. (February 2009)
- [8] J. Sebestyénová, *Case-based Reasoning in Agent-based Decision Support System*, in Acta Polytechnica Hungarica 4(1) (2007)
- [9] K. Erol, J. Hendler, Dana S. Nau, *Semantics for Hierarchical Task-Network Planning*, Technical Report CS-TR-3239/UMIACS-TR-94-3/ISR-TR-95-9, Institute for Advanced Computer Studies, Computer Science Department, University of Maryland (1994)

Acknowledgements

The work reported in this paper was funded by the Systems Engineering for Autonomous