An Approach to Team Programming with Markup for Operator Interaction

(Demonstration)

Nathan Brooks
Carnegie Mellon University
nbb@cs.cmu.edu

Ewart de Visser
Perceptronics Solutions, Inc.
edevisser@percsolutions.com

Timur Chabuk
Perceptronics Solutions, Inc.
timc@percsolutions.com

Elan Freedy
Perceptronics Solutions, Inc.
elanf@percsolutions.com

Paul Scerri
Carnegie Mellon University
pscerri@cs.cmu.edu

ABSTRACT

This paper presents a team plan specification language that combines work in the creation of generic team plans and design of intelligent interfaces. Two key motivations for developing the language are (1) to combine inter-agent cooperation and operator interaction of complex behaviors into a single plan, and (2) to separate plan design and UI design such that they are created by application domain experts and human interaction experts, respectively. The presented result is a generic language for multi-robot plans that defines tasks to be performed, operator interactions for maintaining situational awareness, and mixed initiative actions to react to operator workload.

Categories and Subject Descriptors:
I.2.9 [**Robotics**]: Operator interfaces

Keywords: Robotic agent languages and middleware for robot systems; Intelligence for human-robot interaction; Robot teams, multi-robot systems

1 Introduction

Many exciting, emerging applications of robots involve multiple robots working together on complex tasks under the control of a single human. In such domains, the robots will need to execute sequences of activities in parallel with contingencies, runtime options, and other deviations from a simple linear progression of tasks. A human operator is involved to oversee the execution and is responsible for choosing options, approving actions, making go/no-go decisions, assisting robots in difficult positions, and selecting strategies.

Substantial research effort has looked at how to have cooperative robots controlled by a single operator. Parasuraman[5], Arkin[2], Tambe[6], and others have shown the potential to have some representation of a team plan be invoked by a human and executed by the team. The infrastructure for executing plans can be created once and the team empowered to do different things by developing new plans. Ziparo[9] and Xu[8] describe multi-robot plans using Petri Nets, but the plans do not consider human interaction.

From the perspective of the human, researchers such as Cummings[1], Lewis[7], and Goodrich[3] have shown how intelligent graphical user interfaces (UIs) can dramatically increase a human's ability to understand and interact with robot teams. These utilize domain specific strategies for maintaining operator situational awareness and defining mixed initiative adjustable autonomy parameters, allowing larger and more sophisticated teams to be used effectively.

While these interfaces and others incorporate mixed initiative adjustable autonomy and adapt based on operator workload, they do not adapt based on what plans are being executed. This is undesirable in a number of situations, such as when different instances have different levels of importance and when different instances should have small variations in presentation, options, and default values. To address this, the presented language also includes Situational Awareness and Mixed Initiative (SAMI) markup that provides details about what information should be presented to the human operator and how important it is.

Document approved for public release 88ABW-2012-6315 (3 Dec 2012)

Appears in: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013)*, Ito, Jonker, Gini, and Shehory (eds.), May, 6–10, 2013, Saint Paul, Minnesota, USA.

2 Petri Net Overview

The language is built around Petri Nets, which consist of *places* connected to *transitions*, similar to states and transitions in state machines. However, the Petri Net has a variable number of *tokens* that move from place to place when transitions are fired, the locations of the tokens jointly representing the state. The *edges* between places and transitions specify the number of tokens that must be present in the place for the transition to trigger (in the case of an "incoming" edge from a place to a transition) or the number of tokens that will be put in the place when the transition triggers (in the case of an "outgoing" edge from a transition to a place). The capability of having transitions with multiple incoming and outgoing edges and multiple tokens allows for more representational power than state machines and allows parallelism, synchronization and even counters to be implemented compactly. Below are further extensions.

2.1 Events

These Petri Net models interact with the environment through input and output events. Input events, received as a result of an action taken by some part of the system, can be placed on transitions as a trigger requirement, in addition to the token requirements on the incoming edges. Output events, which represent commands or requests, are added to places and are executed when new, relevant tokens enter the place.

Parameters: When a plan is being developed, there will typically be details for output events that are intended to be provided at runtime. For example, a plan for searching an area will need to know the area's dimensions, but defining those should occur when the operator starts the plan. *Parameters* allow for the design of a generic plan with the details filled in at runtime.

Variables: During the execution of a plan, team members may provide information needed by future events in the plan. For example, a robot might find a location of interest that should be given to a path planner. This is handled with *variables*, which allow the mapping of some component of an input event to components of one or more output events.

2.2 Specialized Tokens

In traditional Petri Nets, all tokens are equivalent and anonymous. A useful extension is colored Petri Nets, which allow individual tokens to be associated with data useful to the plan. The SAMI plan formulation makes use of several specialized tokens which each contain different kinds of data.

Generic tokens are traditional tokens and do not have unique identities or data attached to them.

Proxy tokens contain the name of a robot's proxy and are used to invoke and receive environmental actions on a proxy basis.

Task tokens contain the name of a task and the proxy (if any) that will perform it and are used to invoke and receive environmental actions on a task basis.

2.3 Contingencies

Models based on Petri Nets are able to run multiple plans in parallel, allowing for contingency plans that recognize and react to unexpected behavior or variables beyond the scope of the currently executing plan. Typical problems which would be addressed by contingency plans include low power levels, sensor failures, and reacting to members of the team plan aborting their tasks.

2.4 SAMI Markup

To allow interfaces to dynamically adjust to plans, several types of SAMI markup can be added to the plans at development time. An important concept for the interface is that it does not need to be known when the plans are designed and can be developed independently of the plans, provided it can interpret the SAMI markup.

Information Markup: The first type of information markup tells the UI about environmental events and robot actions that the operator should be aware of. The second type of information markup tells the UI about the progress of the overall plan, e.g., phases of the plan or that a contingency is being invoked. These messages allow the UI to keep the human aware of the progress of the plan. Each piece of information markup can also be associated with specific objects or areas in the environment to allow the UI to be intelligent about the presentation of the messages.

Directives Markup: Directives can give more general situational awareness help than information messages. For example, a plan developer might understand that it will be useful for the human to have a zoomed out map view or a particular robot's video feed at a particular point in the plan. Allowing the designer to embed this sort of information in the plan allows the interface to help the human maintain situational awareness, a critical aspect of effective control[4].

Options Markup: Options markup can modify the set of options presented to the human and specify *mixed-initiative* parameters that are suited to that particular point in the plan. The first type of mixed-initiative parameters controls the number of options requested from services, with the amount chosen by the plan developer to suit the situation and the likely state of the human at that point in time. The options markup can also tell the UI how to handle taking autonomous action.

Importance Markup: The importance markup is used with any of the messages to the UI to describe the importance of the message relative to other things. The importance markup uses a temporal function to indicate to the UI how the importance will change over time. The markup can specify a set of assets for which it applies, so that when the human is micro-managing a robot, only the messages important to that robot can be displayed.

3 Video Submission

The video demonstration displays the execution of a multi-robot SAMI plan, leveraging many of the language's features including task allocation and path planning sub plans, synchronized task execution, and contingency recovery.

4 References

[1] A. S. Macbeth J. C. Clare and M. L. Cummings. Mixed-initiative strategies for real-time scheduling of multiple unmanned vehicles. In *American Control Conf.*, 2012.

[2] Y. Endo, D.C. MacKenzie, and R.C. Arkin. Usability evaluation of high-level user assistance for robot mission specification. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 34(2):168–180, 2004.

[3] M.A. Goodrich, D.R. Olsen, J.W. Crandall, and T.J. Palmer. Experiments in adjustable autonomy. In *Proc. of IJCAI Workshop on Autonomy, Delegation and Control: Interacting with Intelligent Agents*, pages 1624–1629, 2001.

[4] D. Kaber and M. Endsley. The effects of level of automation and adaptive automation on human performance, situation awareness and workload in a dynamic control task. *Theoretical Issues in Ergonomics Science*, 5(2):113–153, 2004.

[5] CA Miller and R Parasuraman. Designing for flexible interaction between humans and automation: Delegation interfaces for supervisory control. *Human Factors*, 49(1):57–75, 2007.

[6] M. Tambe. Agent architectures for flexible, practical teamwork. In *Natl. Conf. on Artificial Intelligence. In Proc. of the*, 1997.

[7] H. Wang, A. Kolling, N. Brooks, S. Owens, S. Abedin, P. Scerri, P. Lee, S. Chien, M. Lewis, and K. Sycara. Scalable target detection for large robot teams. In *Proc. of the 6th Intl. Conf. on Human-robot Interaction*, HRI '11, pages 363–370, 2011.

[8] D. Xu, R. Volz, T. Ioerger, and J. Yen. Modeling and verifying multi-agent behaviors using predicate/transition nets. In *ACM Intl. Conf. Proc. Series*, volume 27, pages 193–200, 2002.

[9] V. A. Ziparo, L. Iocchi, D. Nardi, P. F. Palamara, and H. Costelha. Petri net plans: a formal model for representation and execution of multi-robot plans. In *Proc. of the 7th Intl. Joint Conf. on Autonomous Agents and Multi-Agent Systems*, AAMAS '08, pages 79–86, 2008.