AUTONOMY ARCHITECTURES FOR A CONSTELLATION OF SPACECRAFT

Anthony Barrett

Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Drive, M/S 126-347, Pasadena, CA 91109-8099
phone: +1 818 393-5372, fax: +1 818 393-5244, e-mail: anthony.barrett@jpl.nasa.gov

ABSTRACT

This paper describes three autonomy architectures for a system that continuously plans to control a fleet of spacecraft using collective mission goals instead of goals or command sequences for each spacecraft. A fleet of self-commanding spacecraft would autonomously coordinate itself to satisfy high level science and engineering goals in a changing partially-understood environment – making feasible the operation of tens or even a hundred spacecraft (such as for interferometer or magnetospheric constellation missions).

1. INTRODUCTION

Until the past 5 years, missions typically involved fairly large expensive spacecraft. Such missions have primarily favored using older proven technologies over more recently developed ones, and humans controlled spacecraft by manually generating detailed command sequences with low-level tools and then transmitting the sequences for subsequent execution on a spacecraft controller.

This approach toward controlling a spacecraft has worked spectacularly on previous NASA missions, but it has limitations deriving from communications restrictions – scheduling time to communicate with a particular spacecraft involves competing with other projects due to the limited number of deep space network antennae. This implies that a spacecraft can spend a long time just waiting whenever a command sequence fails. This is one reason why the New Millennium program has an objective to migrate parts of mission control tasks onboard a spacecraft to reduce wait time by making spacecraft more robust [Muscettola et al. 97]. The migrated software is called a “remote agent” and can be partitioned into 4 components:

- a mission manager to generate the high level goals,
- a planner/scheduler to turn goals into activities while reasoning about future expected situations,
- an executive/diagnostician to initiate and maintain activities while interpreting sensed events through reasoning about past and present situations, and
- a conventional reactive controller to interface with the spacecraft to implement an activity’s primitive actions.

In addition to needing remote planning and execution for isolated spacecraft, a trend toward multiple-spacecraft missions points to the need for remote distributed planning and execution. The past few years have seen missions with growing numbers of probes. Pathfinder has its rover (Sojourner), Cassini has its lander (Huygens), Cluster II has 4 spacecraft for multi-point magnetosphere plasma measurements. This trend is expected to continue to progressively larger fleets. For example, one proposed interferometer mission [Mettler&Milman 96] would have 18 spacecraft flying in formation in order to detect earth-sized planets orbiting other stars. Another proposed mission involves 5 to 500 spacecraft in Earth orbit to measure global phenomena within the magnetosphere.

To describe the 4 software components of autonomous spacecraft and constellations, the next section describes a master/slave approach toward autonomously controlling constellations. While being a conceptually simple extension to single-spacecraft autonomy, this approach has several problems that motivate the next section on teamwork. Teamwork replaces masters and slaves with leaders and followers, where a follower has the autonomy to look after its teammates. The fourth section discusses ways to expand teamwork to let each spacecraft function both as a leader and a follower, and the last section concludes by discussing hybrids of the three architectures.

2. MASTER/SLAVE COORDINATION

The easiest way to adapt autonomous spacecraft research to controlling constellations involves treating the constellation as a single spacecraft. Here one spacecraft directly controls the others as if they were connected. The controlling “master” spacecraft performs all autonomy reasoning while the slaves only transmit sensor values to the master and forward control signals received from the master to their appropriate local devices (fig. 1). The executive/diagnostician starts actions and the master’s reactive controller manages actions either locally or remotely through a slave.

The 3 modules above the reactive controller essentially follow the standard belief-desire-intention (BDI) framework [Rao&Georgeff 95]. The mission manager takes a set of *beliefs* and generates *desires* (goals) for the

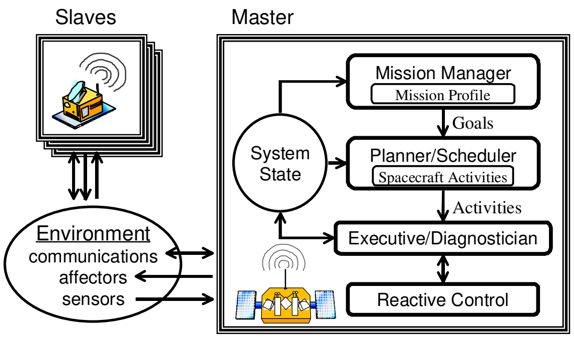

FIG.: 1 Architecture for Master/Slave Coordination

planner/scheduler, which translates them into *intentions* (plans) for execution. Gat describes 3T [Gat 97], another architecture with three layers to deliberate, sequence, and control. While deliberation combines mission management and planning, the other two layers match the executive and the reactive controller. EVAR [Schoppers 95] illustrates another case where the executive subsumes both the planner and mission manager. In general, we can describe most autonomous agent research as variants on the BDI model with different approaches toward implementing the modules and their interactions.

2.1. EXECUTIVE/DIAGNOSTICIAN

At the lowest level the executive/diagnostician (or just "executive") takes an activity sequence, incrementally feeds activities to the reactive controller and monitors results to update the system's state – a model of the constellation and its environment. Since performing an activity might have unintended situation dependant results, blindly feeding primitive activities to the reactive controller is unreliable. The issue here is that the Executive must rapidly diagnose and respond to detected contingencies.

EVAR [Schoppers 95] resolved the problem by compiling large sequences into *universal plans* – a clever encoding of state/response rules that enumerates all states and their appropriate responses. Unfortunately this approach only works in restricted domains where we can make a practical representation that implicitly enumerates all states.

Another approach involves robustly implementing each activity as a *reactive action procedure* (RAP) – an encoding of state/response rules for *anticipated* states [Firby 87]. Here activities fail when the current state falls outside the anticipated set, and failure forces the executive to abort the sequence and inform the planner. The issue now involves how many actions to feed the executive at a time.

For instance, one system uses variable size planning windows to generate sequences where one activity is to plan for the next window [Pell et al. 97], and another system runs the planner continuously and feeds individual activities to the executive as they become executable [Ambrose-Ingerson&Steel 88]. While these examples show that the planner's continual operation is optional, all systems must continually run the executive to actively monitor and diagnose the reactive controllers. This involves using a production system to appropriately apply state/response rules to affect the system state or reactive controller.

2.2. PLANNER/SCHEDULER

While the executive reasons about current and past activities, the planner/scheduler (or just "planner") reasons about future command sequences. Given the heavy use of time and metric resources in spacecraft planning domains, we use a heuristic iterative-repair strategy [Rabideau 99] towards building and maintaining command sequences. This approach takes a complete plan at some level of abstraction and manipulates its actions to repair problems detected by envisioning how the plan would execute on the spacecraft. One type of problem involves multiple simultaneous actions with conflicting resource needs. For example, simultaneously activating too many sensors might cause a bus fault by drawing too much power. Repairing this problem would involve either deleting or moving sensor activation activities in the plan.

At any given moment the mission manager can suggest tasks for the planner/scheduler to add to the constellation's future behavior. Since these tasks are often abstract and might conflict with other established tasks the scheduler continuously debugs its tasks and sends actions to the executive (fig. 2). The planner essentially maintains a set of tasks that are abstract in the far future and become progressively more detailed as their execution times approach. For example, a suggested task to take a picture of a target might involve slewing and possibly calibrating the camera prior to acquiring the image. This task is detailed as its execution time approaches. By continuously detailing the earliest tasks, the planner assures that it always has actions to send to the executive.

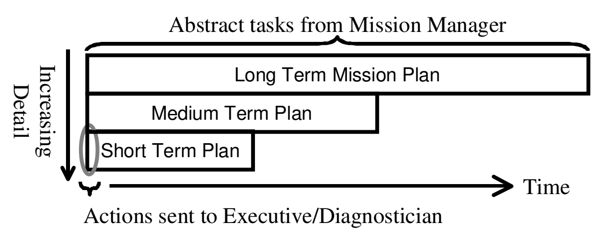

FIG.: 2 Continually updating the spacecraft acitivities

As time progresses, activities move from the future plan through current execution into the past. During this process an activity's expected outcomes get replaced with its sensed outcomes, and the constellation's actual state will drift from the expected state and cause future expectations to drift as well. The planner repairs the tasks whenever this drift causes a conflict.

2.3. MISSION MANAGER

This module facilitates high-level spacecraft commanding by maintaining beliefs involving the high-level mission profile. This profile contains a high level behavioral description for the spacecraft. This description can take many forms from a simple set of temporally constrained goals to an elaborate production system that asserts goals upon detecting user specified scientific opportunities by analyzing parts of the constellation & environment model.

For instance, the spacecraft would have periodic goals to transmit data to Earth. These goals would be temporally constrained in order to synchronize with a ground station. They also have to be high level to determine how to communicate based on the specific state of the spacecraft prior to preparing for a downlink. As another example, the mission manager might apply a feature detection algorithm on a previously captured picture and generate observation goals based on the results.

While a spacecraft can operate entirely autonomously with a mission profile. Humans analyzing the science results will tend to suggest changes to mission goals for answering questions arising from their analysis. We can even vary the constellation's level of autonomy by varying the abstractness of the mission profile. When using primitive action sequences, the profile can short-circuit the planner to allow absolute commanding. Adding abstract tasks to the profile lets the spacecraft adapt its behavior to its local environment, and adding data analysis for rule based autonomous goal generation makes a spacecraft detect and respond to scientific opportunities.

3. TEAMWORK

While the master/slave approach benefits from conceptual simplicity, it relies on an assumption that the master space-craft's reactive controller can continuously monitor the slaves' hardware, and this relies on high-bandwidth highly-reliable communications. Since unintended results occur fairly rarely, one way to relax the bandwidth requirements involves putting reactive controllers on the slaves and only monitoring unexpected events. Unfortunately, this disables the ability to monitor for unexpected events between spacecraft and leads to a host of coordination problems among the slaves [Tambe 97]. Also, failures in the communications system can result in losing slaves.

We can apply teamwork models [Tambe 97, Stone& Veloso 98] to reduce the communications problem by giving the slaves their own executives (fig. 3). This replaces the master/slaves relationship with one between a

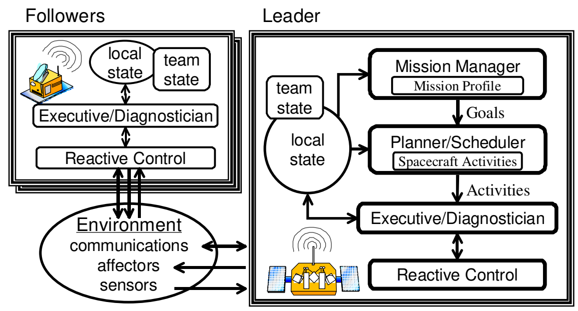

FIG.: 3 Architecture for Teamwork

team leader and its followers. Here each follower can monitor its own performance and selectively transmit results to the leader. Partitioning the system's state into local spacecraft states and shared team-states facilitates this selective transmission. While the spacecraft keep their local states private, they communicate to keep team-states consistent across teams in the constellation.

3.1. REPRESENTING TEAM PLANS

Instead of sending separate actions to each follower for execution, the leader broadcasts the entire reactive team plan[1] to all followers. This lets each follower actively monitor its own progress and passively track its teammates' activities. This passive monitoring process maintains robustness while reducing communications.

In addition to regular activities found in the master/slave approach, reactive team plans also include *team activities*. These define coordination points where the team synchronizes before and after executing the team activity. For instance, a 3 spacecraft interferometer has a combiner spacecraft to generate pictures by processing light reflected from two collector spacecraft. A reactive team plan to control the constellation might have 3 team activities (fig. 4) to coordinate the 3 spacecraft while making an observation, and each activity has 2 or 3 sub-activities defining how the constellation behaves during the joint activities. As illustrated, team activities have brackets and those suffixed with an asterisk only apply to subsets of the team. In this case the subset denotes the combiner spacecraft. The activities in this plan subsequently make the constellation attain a rough formation, dress up the formation for finer tolerances to make a measurement, and transmit the results to Earth.

While this interferometer's impoverished number of spacecraft do not sufficiently motivate the need for teamwork, other interferometer mission proposals describe over a dozen, or even a hundred, collectors to support the combiner. To support teamwork for these larger missions,

[1] Given our heavy use of Tambe's formalism, we adopt his terminology and call a sequence a *reactive team plan*.

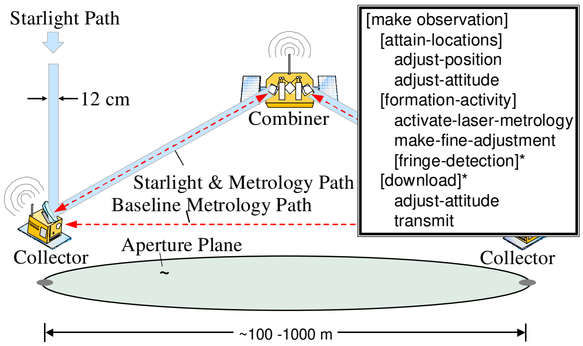

FIG.: 4 Structure of a reactive team plan for a 3 spacecraft interferometer

we must alter the executives' underlying architectures to manage each spacecraft team's associated team-state. We illuminate these changes by describing the machinery underlying team activity execution.

3.2. EXECUTING TEAM ACTIVITIES

A team of spacecraft contains a leader and one or more followers that jointly intend to accomplish some task by executing a team activity. Teams dynamically form when team activity execution starts and dissolve upon completion. When a team performs a task, it shares a team-state. This state contains facts like a list of teammates, their roles in performing the joint task, and other information to coordinate team activity.

Depending on the action, execution can manipulate the reactive controller and alter the local and team-state information. Since team-states are replicated across all teammates, a spacecraft must broadcast all team-state changes to maintain consistency. The standard protocol for changing a team-state is a 3-step process where one spacecraft broadcasts the change, all teammates broadcast acknowledgements in turn, and all teammates update their copies upon hearing everyone else. If a teammate does not respond before a time-out interval, the original spacecraft rebroadcasts the change.

While only transmitting team-state changes reduce communications, the number of broadcasts still implies bandwidth problems as the spacecraft population increases. Stopping spacecraft from broadcasting a change when teammates can infer it from observation further reduces communications [Huber&Durfee 95, Tambe 97]. For instance, the combiner in our interferometer example does not have to signal the end of a formation activity. The mere act of slewing to downlink the results tells the collectors that the formation activity is over.

3.3. GENERATING AND REPAIRING TEAM PLANS

Although reactive team plans might look like an extension on standard hierarchical plans by virtue of the bracket syntax, techniques for building and managing hierarchical plans, like those described earlier, also apply to generating reactive team plans. As such planning does not change much when moving from master/slave plans to reactive team plans. Just like in master/slave coordination, there is a spectrum of ways to generate plans and feed them to the executives. At one extreme the lead spacecraft can generate a whole plan and then feed the resultant sequence to its executives, and at the other extreme it repairs the plan incrementally and maintains a copy in the shared team-state.

The real difference between the two approaches involves limiting the knowledge to plan from. Where the master knew everything about the constellation, the team leader only knows a subset of everything. The issue now becomes a matter of what status information to put in the subset and how fresh to keep it. While increasing the information and its freshness improves the leader's results, it also increases the communication overhead as the constellation's status changes.

A second issue involves whether the information belongs in the team-state, and whether it should be transmitted privately to the leader. While putting information in the team-state increases the followers' abilities to keep track of each other, it also increases the communications overhead. Where changing the team-state involves a broadcast followed by waiting for multiple acknowledgements, changing the leader's local state involves one transmission followed by waiting for the leader's acknowledgement.

One planning approach has the leader managing the team plan and follower roles in the team-state, but lets the followers privately transmit state updates to the leader. Here the leader changes the team plan and roles based on projecting its expected results given the privately received status information.

Another approach still has the leader managing the team plan's activities with heuristically assigned roles in the team-state, but followers keep status information local and submit change requests as they perform their roles in the evolving team plan [Fujita&Lesser 96]. While we can assign and reassign roles at random, a better approach involves auctioning off the unassigned roles to the teammates. The teammates bid on these roles based on local information as well as currently assigned roles, and the leader can either change the plan or assign roles based on these bids.

4. PEER-TO-PEER COORDINATION

The approach to alter communication overhead by distributing execution monitoring across the constellation can extend to also distributing the planning process. This addresses the possibility where the lead spacecraft is disabled. For interferometers this is not an issue because losing the combiner spacecraft ends the mission anyway, but missions like a 50 satellite constellation are functionally redundant and should not end when any one spacecraft is disabled.

One way to increase robustness involves giving the other spacecraft backup planners and mission managers (fig. 5). While this lets the next spacecraft in a designated chain of command replace a disabled leader, these extra modules are underutilized. Instead of transmitting data to a central spacecraft for planning, we can use the extra planners to move parts of the planning process closer to the data. This makes the spacecraft symmetric and coordination becomes a collaborative effort among peers.

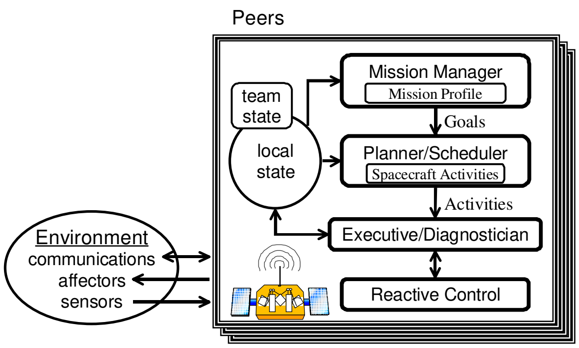

FIG.: 5 Architecture for Peer-To-Peer Coordination

This architecture works particularly well with constellations of satellites that loosely coordinate. For instance, a constellation of picture taking satellites might coordinate to partition desired targets, but each satellite runs in isolation to take its picture. Here the mission managers coordinate to partition the goals, and the planners and executives run in isolation. This class of loose coordination problem is common in the mobile robot community, and some systems even call this module a cooperative planning (or social) module [Müller 96].

4.1. LEVELS OF AUTONOMY

In teamwork or a chain of command, one spacecraft plans how to perform a task and its followers accept and execute the results. Combining loose coordination with teamwork facilitates letting different spacecraft act as leaders for different tasks. Here all spacecraft know about all tasks, and each task has a designated lead spacecraft. Research on autonomy levels [Martin&Barber 96] generalizes this idea. We can give each spacecraft a copy of the plan with tasks annotated with one of 5 autonomy levels:

- Observer: spacecraft does not participate,
- Command-driven: spacecraft serves as a follower,
- Consensus: spacecraft collaboratively plans with others,
- Local: spacecraft plans to perform task alone, and
- Master: spacecraft plans and serves as a leader.

As the 5 definitions imply, autonomy levels specify whether or not a spacecraft can change a task. For instance, a team's leader has tasks annotated with "master", and its followers' tasks have "command-driven" annotations. Given these annotations, a spacecraft can simultaneously serve as a leader and a follower in two separate teams. A spacecraft can even plan and perform tasks in isolation while participating in teams.

While autonomy levels specify which constellation members plan out mission manager requested tasks. These levels are not static – a spacecraft can communicate with the constellation to change a task's autonomy level annotations. For instance, a mission manager might always assign tasks to its spacecraft at the "local" autonomy level. If a team is needed to perform the task, the spacecraft will have to change the annotation to "master." As Martin points out [Martin&Barber 96], this change involves communicating to find spacecraft willing to accept "command-driven" annotations.

Using autonomy levels, we can treat the plan and state information as a shared database where each spacecraft has varying capabilities to modify tasks based on their autonomy-level annotations. Softening the distribution requirement from full to partial plan sharing makes a constellation operate as a team at one point and as multiple independent spacecraft as another. The change involves letting spacecraft keep locally planned and executed tasks private.

4.3. COLLABORATIVE PLANNING

Unlike the other annotations where a single spacecraft plans a task, the "consensus" annotation implies that multiple spacecraft collaboratively plan to perform a task. Collaborative planning involves distributing the plan across the constellation and letting each spacecraft detect and repair problems. The question now becomes a matter of how to keep the plan consistent across the constellation while all spacecraft are updating it. The main objective is to minimize communications overhead while planning.

One approach would fragment the plan and distribute the fragments [Corkill 79]. Since the fragments are disjoint, their union would be consistent. Each spacecraft would expand its own fragment and communicate to detect and resolve interactions. To detect interactions, each spacecraft broadcasts its fragment's effects upon determining them. When a spacecraft hears of an effect that either helps or hinders its own fragment, it initiates a dialog with the broadcasting spacecraft to add signaling actions to their plans to coordinate the interaction. Thus the required bandwidth depends the amount of interaction.

An alternative approach would give every spacecraft a copy of the plan and have them maintain consistency by broadcasting changes as they make them. The main problem with this approach involves communication overhead – the spacecraft would spend most of their time responding to each other's updates.

These two approaches define a whole spectrum of collaborative planners depending on the amount of shared plan and state information. While the first case shared all state information in the form of advertised effects the second shared all plan information.

5. CONCLUSIONS

This paper described several autonomy architectures for an autonomous constellation of spacecraft. Such a constellation would continually plan to control its spacecraft using collective mission goals instead of goals or command sequences for each spacecraft. The first architecture made use of research relating to a single autonomous spacecraft by treated the constellation as a single master spacecraft with virtually connected slaves.

The utilized research describes implementations in terms of 4 interacting modules, and the master/slave architecture placed all modules on the master. While the teamwork and peer-to-peer architectures keep the 4 modules, they progressively give the slaves more authority by replicating more of the modules across the constellation.

While this paper described each architecture in isolation, these architectures can coexist within a constellation. Such a constellation would have 3 classes of spacecraft: leaders, followers, and slaves. Where leaders have the ability to plan and collaborate, followers can only execute plans and watch out for each other. Both leaders and followers can have virtually attached slave spacecraft.

ACKNOWLEDGEMENTS

This work was performed by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

REFERENCES

[Ambrose-Ingerson&Steel 88] J. Ambrose-Ingerson and S. Steel, "Integrated Planning, Execution and Monitoring," Proceedings of AAAI-88.

[Corkill 79] D. Corkill, "Hierarchical Planning in a Distributed Environment," In Proceedings of IJCAI-79.

[Firby 87] R. Firby, "An Investigation into Reactive Planning in Complex Domains," Proceedings of AAAI-87.

[Fujita&Lesser 96] S. Fujita and V. Lesser, "Centralized Task Distribution in the Presence of Uncertainty and Time Deadlines," In Proceedings of ICMAS-96.

[Gat 97] E. Gat, "On Three-Layer Architectures," *Artificial Intelligence and Mobile Robots*, D. Kortenkamp, R. Bonnasso, and R. Murphy, eds., AAAI Press.

[Huber&Durfee 95] M. Huber and E. Durfee, "On Acting Together: Without Communication," AAAI Spring Symposium on Representing Mental States and Mechanisms, 1995.

[Martin&Barber 96] C. Martin and K. Barber, "Multiple, Simultaneous Autonomy Levels for Agent-based Systems," In Proceedings of the Fourth International Conference on Control, Automation, Robotics, and Vision.

[Mettler&Milman 96] E. Mettler and M. Milman. "Space Interferometer Constellation: Formation Maneuvering and Control Architecture," In SPIE Denver '96 Symposium.

[Müller 96] J. Müller, *The Design of Intelligent Agents, A Layered Approach*. Lecture Notes in Artificial Intelligence, Springer-Verlag, 1996.

[Muscettola et al. 97] N. Muscettola, et al. "On-Board Planning for New Millennium Deep Space One Autonomy," Proceedings of IEEE Aerospace Conference 1997.

[Mussliner 93] D. Musliner, *CIRCA: The Cooperative Intelligent Real-Time Control Architecture*, PhD Thesis,University of Michigan, 1993.

[Rabideau et al. 99] G. Rabideau, R. Knight, S. Chien, A. Fukunaga, A. Govindjee, "Iterative Repair Planning for Spacecraft Operations Using the ASPEN System," in Proceedings of iSAIRAS-99.

[Rao&Georgeff 95] A. Rao and M. Georgeff, "BDI Agents: From Theory to Practice," Proceedings of ICMAS-95.

[Pell et al. 97] B. Pell, E. Gat, R. Keesing, N. Muscettola, B. Smith, "Plan Execution for Autonomous Spacecraft," Proceedings of IJCAI-97.

[Schoppers 95] M. Schoppers, "The use of dynamics in an intelligent controller for a space faring rescue robot," *Artificial Intelligence*, 73 (1995) 175-230.

[Simmons 88] R. Simmons, "A Theory of Debugging Plans and Interpretations," In Proceedings AAAI-88.

[Stone&Veloso 98] P. Stone and M. Veloso, "Task Decomposition and Dynamic Role Assignment for Real-Time Strategic Teamwork," Submitted to ICMAS'98.

[Tambe 97] M. Tambe, "Towards Flexible Teamwork," *Journal of Artificial Intelligence Research*, 7:83-124.