Kansas State University
234 Nichols Hall
Manhattan, KS 66506-2302

Phone: (785) 532-6350
Fax: (785) 532-7353
http://macr.cis.ksu.edu/

Integrating Performance Factors into an Organization Model for Better Task Allocation in Multiagent Systems

Christopher Zhong
czhong@k-state.edu

Scott A. DeLoach
sdeloach@k-state.edu

MACR-TR-2010-02

April 2010

Abstract

1 Introduction

As computing systems become more advanced and complex, more is being expected out of them. Two of these expectations are (1) to be able to self-adapt to various failures and (2) to have more integration with humans. Rising expectations on self-adaptive systems include the ability to automatically correct itself in a fluid and dynamic environment (autonomic systems). Multiagent concepts [3] are well-suited for developing self-adaptive systems. Russell and Norvig [20] define agents as able to perceive and act autonomously such that their actions are based on their own experiences rather than predefined knowledge. Multiagent systems exploit this behaviour to self-correct, if an agent should fail, another agent can take over. One area of multiagent research is to leverage organizational concepts to produce organization-based multiagent system such as **O**rganization **M**odel for **A**daptive **C**omputational **S**ystems (OMACS) [5], **O**rganizations **per A**gents (OperA) [6], **O**rganizational **M**odel for **N**ormative **I**nstitutions (OMNI) [7], and HarmonIA [33]. By leveraging organizational concepts, the problem of task allocation can be addressed in a general manner by capturing the necessary information. Various research teams have applied these concepts in robotics, particularly to multirobot systems [2, 8, 9, 16, 25, 27, 30].

Traditionally, humans have been considered as users of a system; they are not typically considered as a part of the internal process of the system. To facilitate better integration with humans, a system should include humans as part of it's internal process. By including humans as part of the system, the system is able to use humans as part of its self-adapting mechanism(s) as well as to allow interaction to occur between various aspect of the system with humans. Due to increased amounts of information to be handled when designing systems that include humans as part system, the complexity often spirals out of control. However, organization-based models are well-suited to facilitate this integration because they already provide a basic framework to begin capturing this information about humans. Furthermore, by leveraging model-driven engineering (MDE), runtime models (commonly referred to as *models@run.time* [23]) can be produced that are able to manage the complexity of dealing with these information at runtime.

The question is what type of information do we want to capture about the human? Since the focus is on facilitating better task allocation algorithms as well as better interactions with humans, information about human performance needs to be captured. There are various factors that can impact performance. Wickens *et al.* [34] have examined and explained a variety of human performance factors for particular tasks. There are a wide variety of factors that affects performance, some are about the well-being of the humans, some are about the tasks, and some are environmental. One way to capture these factors is through **P**erformance **M**oderator **F**unctions (PMFs) [24], particularly human performance moderator functions (HPMFs).

This report proposes a runtime model based on OMACS that can capture performance factors as the first step towards integrating humans as part of the system. The rest of this paper is organized as follows: § 2 highlights the foundational work on which the work in this report is based. § 3 discusses research that are similar to the work in this paper. § 4 talks about the proposed runtime model. § 5 highlights some of the limitations of the proposed model as well as ongoing work in addressing those limitations. And we conclude in § 6 by highlighting the contributions of including performance factors with respect to facilitating better task algorithms as well as better integration with humans.

2 Background

This section highlights three key background areas required to understand this report. § 2.1 highlights the organizational model that captures the ability of agents to carry out as well as facilitating reorganization. § 2.2 highlights the goal model that represents the goal of a system as well as capturing the progress towards that goal. And finally, § 2.3 highlights the mathematical concepts that describes the performance and capabilities with regards to humans.

2.1 OMACS

The **O**rganization **M**odel for **A**daptive **C**omputational **S**ystems (OMACS) [5] is a model for representing adaptive groups that defines the knowledge required to allow reorganization due to failure or changing goals. As shown in Figure 1, an OMACS *organization* consists of *goals*, *roles*, *agents*, and *capabilities*. *Goals* are high-level descriptions of what the system is supposed to accomplish [19]. Conversely, *roles* are high-level specifications on how to achieve specific goals. *Agents* are autonomous entities that can perceive and act within their environment [19]. *Capabilities* represent the notion of an agent's ability to perceive and act.

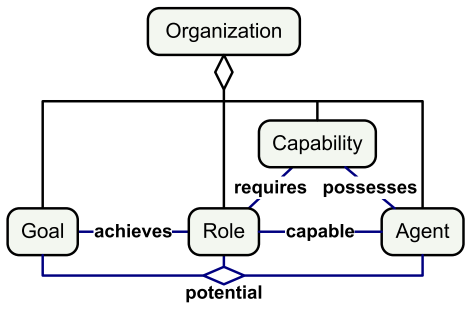

Figure 1. Organization Model

These entities are related to one another via a set of functions: *achieves*, *requires*, *possesses*, *capable*, and *potential*. The *achieves* function defines the effectiveness (0.0 . . . 1.0) of a role in achieving a goal, where 0.0 means that the role is unable to achieve the goal. The *requires* function defines the capabilities that a role needs in order for agents to carry out the role's behavior. The *possesses* function defines the effectiveness (0.0 . . . 1.0) of an agent's capabilities, where 0.0 means that the capability of the agent is broken or non-existent. The *capable* function specifies how well (0.0 . . . 1.0) an agent can perform a role, which is obtained from the *rcf* function. The *rcf* function derives the score by using the *requires* and *possesses* functions. The *potential* function defines how well (0.0 . . . 1.0) an agent can perform a role to achieve a goal, which is also derived from the achieves and capable functions.

OMACS-based systems use the potential function to automatically make assignments[1]. An assignment is a tuple consisting of a single goal, a single role, and a single agent. As the capabilities of agents change throughout the course of a system's execution, these changes are reflected by the possesses function, which then is also reflected by the capable function, and finally by the potential function. Should an agent reach a point where it is no longer capable of performing a role, the capable function would reflect a score of 0.0. This would in turn cause the potential function would reflect a score of 0.0 as well. This triggers the reorganization mechanism to replace the failed agent. This process is how an OMACS-based systems adapt to failures.

2.2 GMoDS

The set of goals captured by OMACS is simple at best; it represents the current set of goals that the organization is actively pursuing. A sophisticated model is required in order to represent a more complex set of requirements such as optional goals, alternative goals, goals that becomes pursuable once certain goals are achieved, and goals that come into play if certain conditions are met.

The **G**oal **Mo**del for **D**ynamic **S**ystems (GMoDS) [15] captures a system's requirements as a single goal tree. The top-level or overall goal is decomposed into subgoals that follows the classic AND/OR goal decomposition [32]. If all subgoals must be achieved to achieve the parent goal, then the parent goal is an AND-goal. Conversely, a goal is an OR-goal if that goal is achieved when any of it's subgoals are achieved. At the lowest level of the goal tree are the leaf goals, which are goals that are used by OMACS for making assignments.

In addition, GMoDS provides two extensions to the classic AND/OR goal tree; (1) a sequential ordering for achieving goals through the *precedes* relation, and (2) a *triggers* relation for specifying events that causes the

[1]Making an assignment is the term used to mean allocating a task.

creation of new goals or removal of existing goals. If goal A *precedes* goal B, then goal A must be completed first before goal B can be attempted. If goal A *triggers* goal B with event E, then while in the pursuit of goal A, goal B is created every time event E occurs.

GMoDS provides the needed ability to track progression in achieving a system's overall goal (the goal tree), the ability to dynamically adapt to variations in system parameters (the *triggers* relations), and the ability to systematically make incremental progress towards achieving the system's overall goal (the *precedes* relation).

2.3 PMF

Human factors sciences is the study and understanding on the capabilities of humans. Human factors engineering is the application of the knowledge about human capabilities to designing better systems. Wickens *et al.* [34] provided a number of design principles and methodologies on how using knowledge can lead to better systems.

Performance **M**oderator **F**unctions (PMFs) quantify the impact of human performance to internal and external stressors. In addition, PMFs help to capture the role of personality and individual differences. A PMF captures a dose-response type of relationship between a performance moderator and the level of performance. These moderators reflect significant dimensions of individual and group differences such as intelligence, skill, judgement, leadership, emotion, organizational culture, motivation, dedication, and slips / lapses / biases. In addition, external stressors on individual and / or groups such as task time, noise, fatigue, stree, and opponent actions.

3 Related Work

In multi-robot systems, Parker [17] states that there are three approaches to task allocation: bioinspired approaches, organizational approaches, and knowledge-based approaches.

In bioinspired approaches, observations on animal behaviors are applied to multi-robot systems. The robots are typically homogeneous and exists in large numbers (i.e., swarms). Individually, each robot possesses very limited capability. However, when they are grouped together in large numbers and interact as a *collective*, a group-level intelligent behaviour emerges. Because it is assumed that every robot has ability to sense the relevant information in their environment (i.e., *stigmergy*), communication among robots is reduced significantly. Even in the situations when *stigmergy* is not available, robots only need to broadcast minimal information about their state or environment. No communication ever occurs about which robots are performing which tasks. A task is allocated when a robot senses that a task needs to be performed and proceeds to perform it. Should a robot fail when performing a task, another robot simply replaces the failed robot. By following this basic behavior, a *collective* of these robots can achieve the overall system goal. Examples of bioinspired systems are Balch and Arking [1]; Kube and Zhang [11]; Kubo and Kakazu [12]; Matarić [13]; McLurkin and Smith [14]; Passino [18]; Stilwell and Bay [26]; Kazuo and Suzuki [28]; and Sun, Lee, and Sim [29].

Organizational approaches utilize organizational theory for task allocation in multi-robot systems. Robots in these systems are typically heterogeneous as they can possess varying capabilities. Within the organizational approaches are two approaches to task allocation: role-based and market-based. Role-based approaches employ the use of roles to divide up the work that needs to be done. A role can consists of one or more tasks that needs to be completed. Robots then select or are assigned the roles which are best suited for them based on their capabilities. Examples of role-based approaches are Simmons *et al.* [25]; and Stone and Veloso [27]. Market-based approaches use principles and theories of market economies to allow each robot to negotiate with other robots on which tasks they should perform. Robots communicate with each other on the cost or utility of tasks and the tasks are then assigned to the robot with the lowest cost or highest utility. Examples of market-based approaches are M+ [2]; Sold! [9]; Sariel [21]; Sariel, Balch, and Erdogan [22]; Zlot and Stentz [36].

Knowledge-based approaches share ontological or semantic information among the robots as the basis for task allocation. Since the robots in knowledge-based approaches are typically heterogeneous, the focus of knowledge-based approaches is allowing robots to easily share information about their capabilities to each other. Various techniques have been applied to sharing the information. COBOS [8] uses a *task suitability matrix* for task allocation. The *task suitability matrix* maintains the *suitability* of each robot for each task. The *suitability* of a robot is computed based on the *intrinsic* abilities of the robot to the task and the *extrinsic* factors of the task. In ALLIANCE [16], every robot contains a model of every other robot. This model contains information about the performance of the robots and tasks-related information. These models are populated through observation. Robots then use these models to determine which task(s) to perform. ASyMTRe [30] and ASyMTRe-D [31] captures the capabilities of robots as *schemas*. *Schemas* are building-block type capabilities with inputs and outputs that are semantic information types. By matching the inputs and outputs of *schemas*, a valid flow can be formed through various *schemas* in completing tasks.

4 Attributes

OMACS [5] is currently unable to capture some types performance factors, particularly performance factors such as *location* and *reaction time*. A number of additions and changes are required to facilitate OMACS to capture these performance factors. Figure 2 shows the proposed additions and changes to the OMACS. Entities are rounded rectangles, the directed lines are functions that link two entities together, and the squared rectangles are the result of the functions. Greyed out elements are existing elements in OMACS. There is one new entity and four new functions.

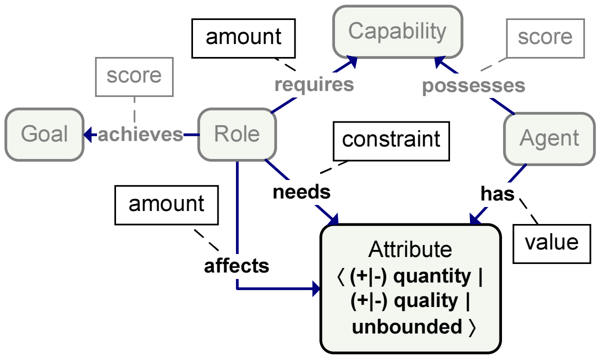

Figure 2. Organization Model

The new entity is called an *attribute*. An *attribute* is a description or characteristic of an object. Currently, there are five types of attributes: positive / negative quality-type, positive / negative quantity-type, and unbounded-type. A quality-type attribute constrains the value to the $0.0 \ldots 1.0$ range, a quantity-type attribute constrains the range to $0.0 \ldots \infty$, and an unbounded-type attribute does not constrains the values. The positive or negative prefix indicates the type of scale used to measure the values in relation to one another. Some attributes can be represented as either positive (the higher the value, the better) or negative (the lower the value, the better). For instance, energy versus fatigue. These two attributes represent the same concept except that for energy the higher value is better, while for fatigue, the lower value is better.

The *has* function specifies a relation between an agent and an *attribute* and returns a value depending on the type of that attribute: quantity $(0.0 \ldots \infty)$, quality $(0.0 \ldots 1.0)$, or unbounded $(-\infty \ldots +\infty)$. Since the *has* function provides a unary relation between a single value of an attribute to an agent, it is straightforward to model more complex characteristics such as *location* as *location* can be broken down into three attributes: *longitude*, *latitude*, and *altitude*. A logical grouping of the three attributes (*longitude*, *latitude*, and *altitude*) into the *location* attribute would not provide any theoretical benefits.

The *affects* function specifies a relation between a role and an *attribute* and returns an amount of the change to the value of the *has* function. The amount captures the changes that are to occur at the completion of the role. This amount can be either a positive (increasing) or negative (decreasing) amount, which is necessary because some changes that increase future performance and some that decrease future performance. For instance, one approach of capturing experience is through *flight time* (the higher the better). After performing a particular role, *flight time* increases (which is generally a good thing) and *fatigue* increases (which is generally a bad thing).

The *needs* function specifies a relation between a role and an *attribute* and returns a set of constraints. The set of constraints specify the requirements an agent needs to meet in order for it to be eligible for assignment. The constraints are necessary because they are needed for making better assignments. Currently, there are four operators for specifying constraints: '$< x$', '$= x$', '$> x$', and '$x - y$'. A series of constraints are separated by ',' operator. For instance, the constraints $(< 10,\ 15 - 17,\ > 20)$ means that any agent with that particular attribute and the value of that attribute is either less than ten, between fifteen and seventeen, or greater than twenty can be assigned to play the role. The constraints specified as a set are purely disjunctive because the values of any attribute represents a single state.

In OMACS, the *capable* function specifies a relation between a role and an agent and returns a value $(0.0 \ldots 1.0)$ of how well an agent can play a role. Here, the *capable* function is redefined such that it reflects two functions: *rcf* and *raf*. The *raf* (role-attribute function) specifies whether an agent has the necessary attribute(s) to play the a role ($true$ or $false$). The return value of the *capable* function is equal to the *rcf* if *raf* is $true$. Otherwise, the return value is 0.0.

4.1 Evaluation

We use the conference management system (CMS) [4, 35] as basis for evaluating the usefulness of the proposed model versus the original model. The CMS represents a conceptual model of the process that takes place leading up to a scientific conference. For instance, the process where authors submit their papers, reviewers are given papers to review, making decisions on whether to accept the papers, and sending the accepted papers for printing. Figure 3 shows the GMoDS model that represents the CMS process, where system goals are represented and further decomposed into subgoals. The top-level goal of the CMS is to *manage submissions*, which is decomposed into six conjunctive subgoals, which are also further decomposed into subgoals. At the bottom of the goal tree are the leaf goals, which are *collect papers*, *distribute papers*, *partition papers*, *full paper*, *short paper*, *poster paper*, *collect reviews*, *make decision*, *inform declined*, *inform accepted*, *collect finals*, *make CD*, and *print proceedings.* In order to set up the scenarios for the experiments, the *review papers* (which was originally a leaf goal) is decomposed into three subgoals: *full paper*, *short paper*, and *poster paper*.

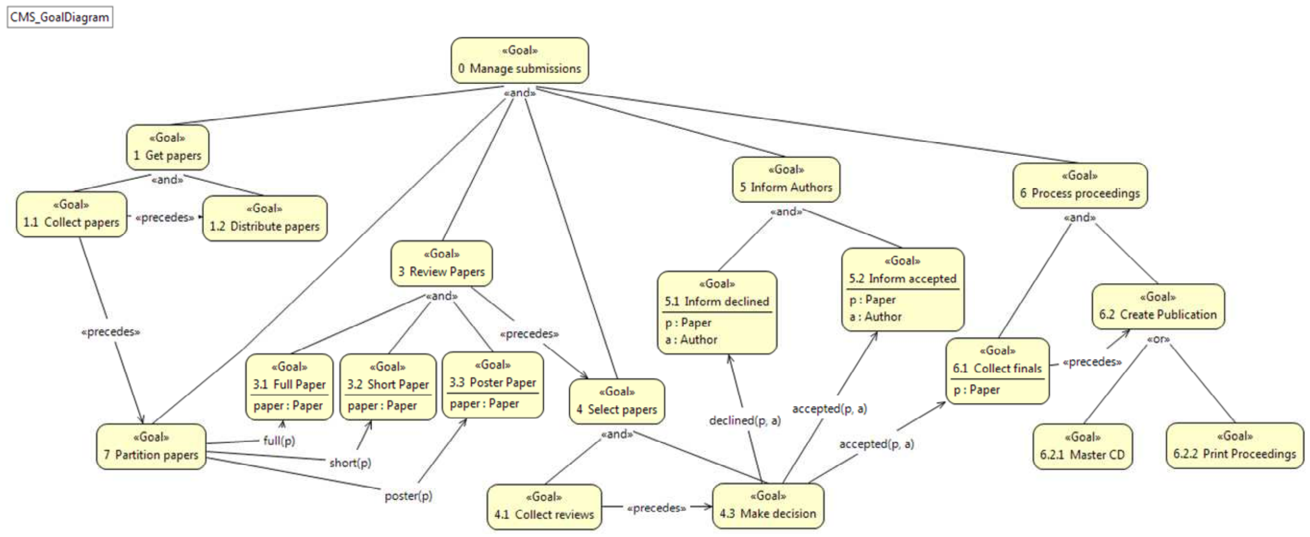

Figure 3. CMS Goal Model

Figure 4 shows the role model where each leaf goal is mapped to a specific role. These roles are required to achieve their associated leaf goal(s). These roles also define what capabilities are required by agents in order to peform them. Similar to the decomposition of the *review papers* goal, the *reviewer* role is also broken down into three subroles: *full reviewer*, *short reviewer*, and *poster reviewer*.

The CMS experiments are set up such that there are a fixed number of 20 reviewers. There are 120 experiments and each experiment have a fixed number of reviews that ranges from $30 - 150$ reviews. In each experiment, we compare four general reorganization algorithms; two algorithms for the original model versus two algorithms for the proposed model. All four algorithms only utilize information that is provided by the models. The two algorithms for the original model are the random and round robin algorithms, and the two algorithms for the proposed model are the greedy and optimal algorithms. Although the goal model depicts the entire process of the CMS, the focus of the experiments are centered on the goal *review papers* and its three subgoals.

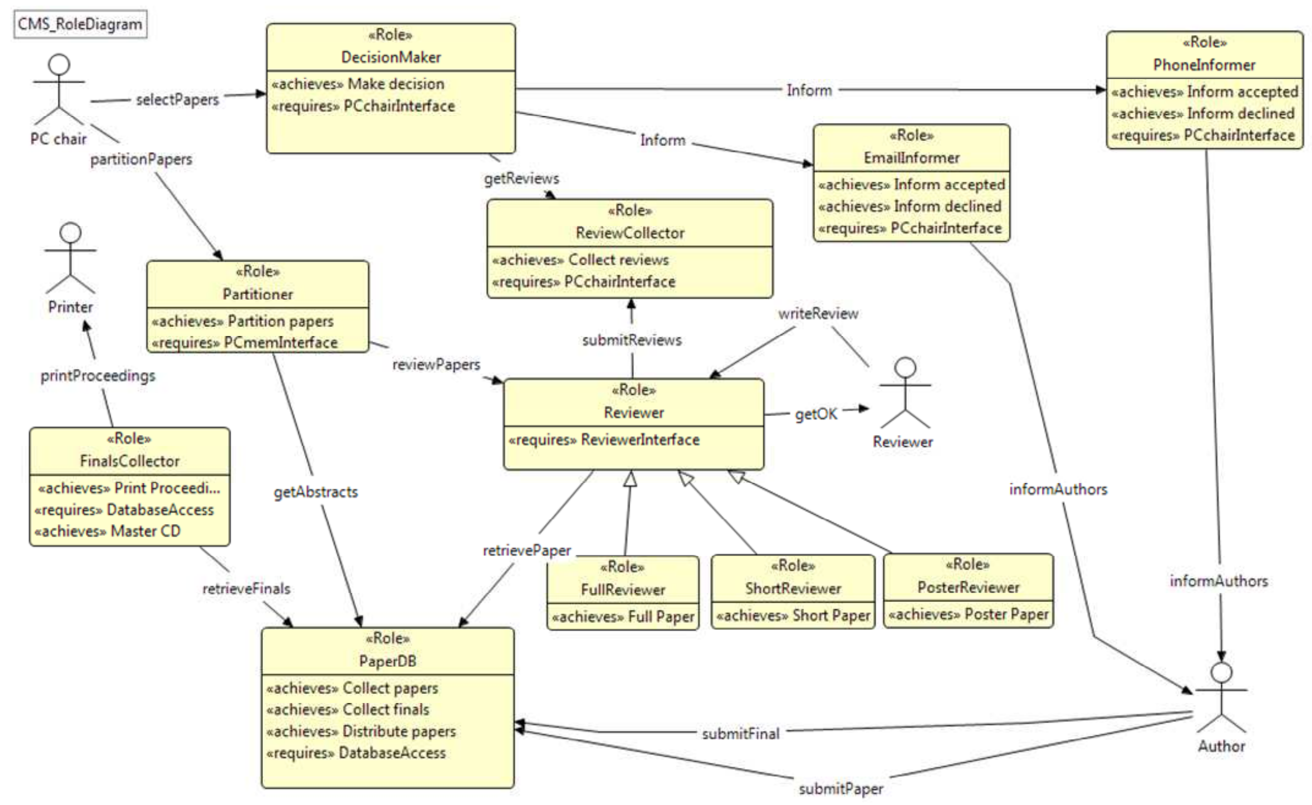

Figure 4. CMS Role Model

4.2 Results

The purpose of the reorganization algorithm is to distribute the reviews (n) to all 20 reviewers such that the average quality of reviews remain as high as possible. The function to compute the quality of reviews is based on the workload of the reviewer. We used the formula $\min\left(\frac{100}{w}, 1\right)$, where w is the workload of the reviewer, to compute the quality of reviews produced by a reviewer. There are three types of papers to be reviewed: full papers, short papers, and poster papers. A full paper adds 40 to workload, short paper adds 20 to workload, and poster paper adds 10 to workload. For any n, the distribution of the types of papers to be reviewed are as follows.

$f = \left\lfloor \frac{n}{3} \right\rfloor$, where f is the number of reviews that are full papers.

$p = \left\lceil \frac{n}{3} \right\rceil$, where p is the number of reviews that are poster papers.

$s = n - f - p$, where s is the number of reviews that are short papers.

The random algorithm randomly selects a reviewer and assigns a paper to that reviewer. The process continues until all papers have been assigned. Because of the random nature of the algorithm, for a given n reviews, the random algorithm may randomly create the optimal assignment set or the worst assignment set. Thus, for every n reviews, the random algorithm is performed $10,000$ times to normalize the results. Because the random algorithm is for the original model, it does not have access to the new functions defined in the proposed model (Figure 2).

The round robin algorithm selects a paper and assigns it to the first reviewer in line (after which, the first reviewer moves to the back of the line), selects the next paper and assigns it to the next reviewer in line, and so forth until all papers have been assigned. The round robin algorithm is affected by the ordering of the papers assigned. A good ordering results in a higher average, while a bad ordering results in a lower average. Because of this, the ordering of papers is randomized and performed $10,000$ times to normalize the results for every n reviews.

Because the round robin algorithm is for the original model, it does not have access to the new functions defined in the proposed model (Figure 2).

The greedy algorithm sorts the list of reviews by their workload from highest to lowest. It then assigns the highest workload paper to the reviewer with the lowest workload. This process continues until all papers have been assigned. This algorithm always produce the same assignment set for a given n and so only one run is required.

The optimal algorithm maximizes the sum in Equation (1). Where w_r is the workload of reviewer r and n_r is the number of reviews by reviewer r. In general, the problem of task allocation is NP-hard [10]. Even with the sum in Equation (1), a general optimal algorithm is still required to search through all possible assignment sets to find the one that maximizes the sum. Because of this, we did not implement the optimal algorithm to obtain the results but rather computed the results by hand. Our analysis indicated that the optimal assignment set involved keeping 19 reviewers at 100 workload or less, and 1 reviewer is assigned to review everything else.

$$\sum_{r=1}^{20} \min(\frac{100}{w_r}, 1) \times n_r \tag{1}$$

Figure 5 shows the results of the four algorithms. There are two points of interest in the graph. The first cut off point of 86 reviews is the point where all reviewers have a workload of 100 or less. After that point, at least one reviewer will have a workload over 100. Because with 86 reviews; 28 of them are full papers, 29 of them are short papers, and 29 of them are poster papers, which gives a combined workload of 1990. We have 20 reviewers with a combined workload of 2000. At 87 reviews, we have 29 full papers, 29 short papers, 29 poster papers, which is a combined workload of 2030. By utilizing information provided by the *affects* function, the greedy algorithm is able to keep the quality of papers at 100% until 86 reviews, whereas the round robin begins to drop somewhere around 45 reviews. Up to this point the random algorithm performed the worst.

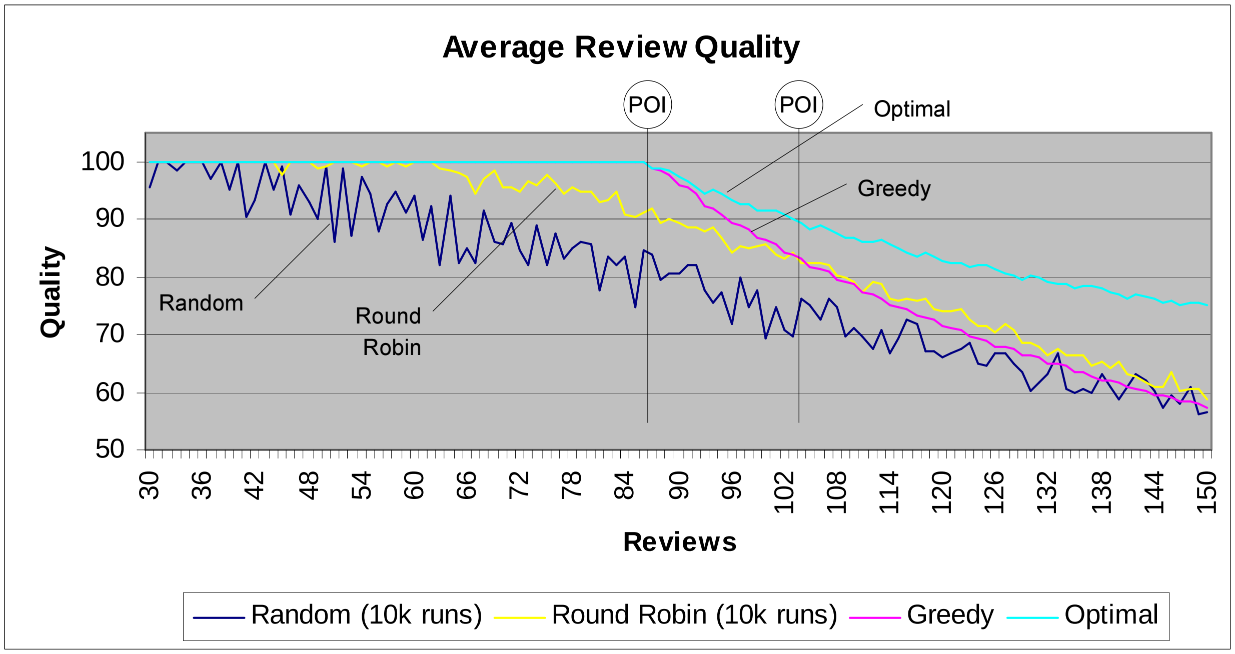

Figure 5. Results

Another point of interest is around the 105 reviews, where the round robin algorithm begins to perform slightly better than the greedy algorithm. Around this point, the greedy algorithm produces an assignment set in which every reviewer has a workload over 100. However, the round robin algorithm is still able to produce assignment sets in which some reviewers have a workload of less than 100. This is situation is highly dependent on the order in which the reviews are handed out; a good ordering produces higher average quality of reviews while a bad ordering produces lower quality of reviews.

5 Future Work

The models shown in Figure 3 and Figure 4 required a change to accommodate the ability to capture workload requirements of the three types of reviews. To accomplish this the *review papers* goal is decomposed into three subgoals, one for each type of review, so that the *reviewer* role can also be decomposed into three subroles. These subroles are then mapped to different workload amounts. This is a limitation of the proposed model of Figure 2, which we are addressing in an updated model as shown in a preliminary version in Figure 6. Ideally, the goal and role model would remain unmodified to capture these type of information, a new model (tentatively called the task model) links tasks (which are role-goal pairs) to *performance factors*, which brings us to the next limitation.

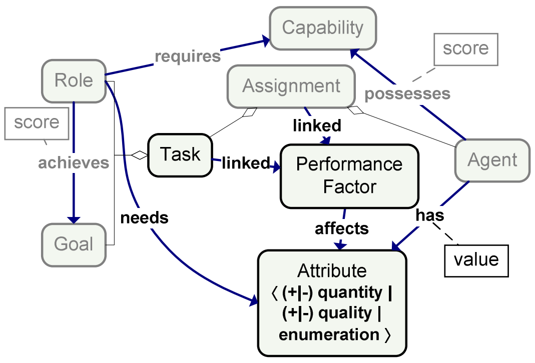

Figure 6. Organization Model

The *affects* function is specified as relation between a role and *attribute* with an amount. This is limiting in three ways; (1) the amount does not take into consideration different goals that can be achieved by the role, (2) the amount does not take into consideration different agents that can perform the role, and (3) changing the amount during runtime requires external mechanism(s). In the next revision of the proposed model, all three limitations are addressed through the *task* and *performance factor* entities. The *task* entity captures the role-goal pairings, while *performance factor* captures PMFs that compute values. This approach presents an unintended benefit of allowing a system to continuously track the status of agents.

The CMS experiments utilize only one attribute. Typically, PMFs are dependant on other PMFs and sometimes the dependencies are circular. Utilizing multiple attributes should provide further insight into how these dependencies can be capture as well as provide some ideas on addressing circular dependencies PMFs. Using multiple attributes introduce the need to know how these values combine together so as to allow task allocation algorithms to select the appropriate agent from a set of possible agents.

Having this information available provides a unique opportunity to predict effect of various tasks. For instance, this information can be used to prevent situations in which the only agent capable of completing a task is assigned to another task, thus rendering it unable to complete that task that only it can. This creates a ripple effect that eventually leads to the system failing to achieve the top-level goal as there are no longer agents left capable of completing the task. Using the information in a predictive fashion is something that will be explored once the proposed model is in a stable state.

6 Conclusions

In conclusion, the proposed model (§ 4) is able to capture information about the state of agent through the *attribute* entity. This is the first step toward including humans as part of the system. The associated functions (*has*, *affects*, and *needs*) provide the structure so that a system is able to utilize the new information for task allocation. The results shown in § 4.2 indicates that general algorithms can benefit from using these information in the task allocation process. Although, there are some limitations to the proposed model, they are actively being addressed for the next revision of the model.

References

[1] T. Balch and R. C. Arkin. Communication in Reactive Multiagent Robotic Systems. *Autonomous Robots*, 1(1):27–52, March 1994.

[2] S. C. Botelho and R. Alami. M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1234–1239. IEEE, 1999.

[3] K. M. Carley and L. Gasser. Computational Organization Theory. In G. Weiss, editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, pages 299–330, Cambridge, MA, USA, 1999. MIT Press.

[4] S. A. DeLoach. Modeling Organizational Rules in the Multi-agent Systems Engineering Methodology. In R. Cohen and B. Spencer, editors, *Advances in Artificial Intelligence: 15th Conference of the Canadian Society for Computational Studies of Intelligence*, volume 2338 of *LNAI*, pages 1–15. Springer-Verlag, 2002.

[5] S. A. DeLoach, W. Oyenan, and E. T. Matson. A Capabilities-based Model for Adaptive Organizations. *Journal of Autonomous Agents and Multi-Agent Systems*, 16(1):13–56, February 2008.

[6] V. Dignum. *A Model for Organizational Interaction: Based on Agents, Founded in Logic.* PhD thesis, Utrecht University, 2004.

[7] V. Dignum, J. Vázquez-Salceda, and F. Dignum. OMNI: Introducing Social Structure, Norms and Ontologies into Agent Organizations. In *PROMAS*, pages 181–198, 2004.

[8] C.-H. Fua and S. S. Ge. COBOS: Cooperative Backoff Adaptive Scheme for Multirobot Task Allocation. In *IEEE Transactions on Robotics*, volume 21, issue 6, pages 1168–1178. IEEE, 2005.

[9] B. P. Gerkey and M. J. Matarić. Sold!: Auction Methods for Multirobot Coordination. In *IEEE Transactions on Robotics and Automation*, volume 18, issue 5, pages 758–768. IEEE, 2002.

[10] B. P. Gerkey and M. J. Matarić. A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems. *The International Journal of Robotics Research*, 23(9):939–954, September 2004.

[11] C. R. Kube and H. Zhang. Collective Robotics: From Social Insects to Robots. *Adaptive Behavior*, 2(2):189–218, September 1993.

[12] M. Kubo and Y. Kakazu. Learning Coordinated Motions in a Competition for Food between Ant Colonies. In *Proceedings of the Third International Conference on Simulation of Adpative Behavior: From Animals to Animats 3*, pages 487–492, Cambridge, MA, USA, 1994. MIT Press.

[13] M. J. Matarić. Designing Emergent Behaviors: From Local Interactions to Collective Intelligence. In *Proceedings of the Third International Conference on Simulation of Adpative Behavior: From Animals to Animats 2*, pages 432–441, Cambridge, MA, USA, 1993. MIT Press.

[14] J. McLurkin and J. Smith. Distributed Algorithms for Dispersion in Indoor Environments Using a Swarm of Autonomous Mobile Robots. In R. Alami, R. Chatila, and H. Asama, editors, *7th International Symposium on Distributed Autonomous Robotic Systems.* Springer, 2004.

[15] M. Miller. A Goal Model for Dynamic Systems. Master's thesis, Kansas State University, April 2007.

[16] L. E. Parker. ALLIANCE: An Architecture for Fault Tolerant Multirobot Cooperation. *IEEE Transactions on Robotics and Automation*, 14(2):220–240, April 1998.

[17] L. E. Parker. Distributed Intelligence: Overview of the Field and its Application in Multi-Robot Systems. *Journal of Physical Agents*, 2(1):5–14, 2008.

[18] K. M. Passino. Biomimicry of Bacterial Foraging for Distributed Optimization and Control. *IEEE Control Systems Magazine*, 22(3):52–67, 2002.

[19] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1995.

[20] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 2nd edition, 2002.

[21] S. Sariel. *An Integrated Planning, Scheduling and Execution Framework for Multi-Robot Cooperation and Coordination.* PhD thesis, Istanbul Technical University, Institute of Science and Technology, Istanbul, Turkey, June 2007.

[22] S. Sariel, T. Balch, and N. Erdogan. Incremental Multi-Robot Task Selection for Resource Constrained and Interrelated Tasks. In *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2314–2319, 2007.

[23] D. C. Schmidt. Guest Editor's Introduction: Model-Driven Engineering. *Computer*, 39(2):25–31, February 2006.

[24] B. G. Silverman. Performance Moderator Functions for Human Behavior Modeling in Military Simulations. `http://www.seas.upenn.edu/~barryg/PMFset.zip`, 2002.

[25] R. Simmons, S. Singh, D. Hershberger, J. Ramos, and T. Smith. First Results in the Coordination of Heterogeneous Robots for Large-Scale Assembly. In *Experimental Robotics VII*, volume 271 of *Lecture Notes in Control and Information Sciences*, pages 323–332. Springer Berlin / Heidelberg, 2001.

[26] D. J. Stilwell and J. S. Bay. Toward the Development of a Material Transport System using Swarms of Ant-like Robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 1, pages 766–771, Atlanta, GA, USA, 1993.

[27] P. Stone and M. Veloso. Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence*, 110(2):241–273, June 1999.

[28] K. Sugihara and I. Suzuki. Distributed Algorithms for Formation of Geometric Patterns with Many Mobile Robots. *Journal of Robotic Systems*, 13(3):127–139, 1996.

[29] S.-J. Sun, D.-W. Lee, and K.-B. Sim. Artificial Immune-Based Swarm Behaviors of Distributed Autonomous Robotic Systems. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 4, pages 3993–3998, 2001.

[30] F. Tang and L. E. Parker. ASyMTRe: Automated Synthesis of Multi-Robot Task Solutions through Software Reconfiguration. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1501–1508. IEEE, April 2005.

[31] F. Tang and L. E. Parker. Distributed multi-robot coalitions through asymtre-d. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 2606–2613. IEEE, August 2005.

[32] A. van Lamsweerde, E. Letier, and R. Darimont. Managing Conflicts in Goal-Driven Requirements Engineering. *IEEE Transactions on Software Engineering*, 24(11):908–926, 1998.

[33] J. Vázquez-Salceda and F. Dignum. Modelling Electronic Organizations. In V. Marik, J. Müller, and M. Pechoucek, editors, *Multi-Agent Systems and Applications III*, volume 2691 of *LNAI*, pages 584–593. Springer-Verlag, 2003.

[34] C. D. Wickens, J. D. Lee, Y. Liu, and S. E. Gordon-Becker. *An Introduction to Human Factors Engineering.* Prentice Hall, 2 edition, 2003.

[35] F. Zambonelli, N. R. Jennings, and M. Wooldridge. Organisational Rules as an Abstraction for the Analysis and Design of Multi-Agent Systems. *International Journal of Software Engineering and Knowledge Engineering*, 11(3):303–328, 2001.

[36] R. Zlot and A. Stentz. Market-based Multirobot Coordination for Complex Tasks. *The International Journal of Robotics Research*, 25(1):73–101, 2006.