

# Scheduling the Activities of Distributed Teams \*

Pedro Szekely, Rajiv Maheswaran, Craig Milo Rogers, Romeo Sanchez  
USC/Information Sciences Institute  
4676 Admiralty Way, Mairna del Rey, CA 90292  
{pszekely, maheswar, rogers, rsanchez}@isi.edu

## ABSTRACT

Project objectives can often be achieved in many different ways. The tasks involved can be done by different people at different locations, can be sub-divided into subtasks in multiple ways, can be done synchronously or asynchronously, collocated or distributed. Different options lead to different timing considerations, risks, costs, durations and quality of outcomes. Allocating people and resources to tasks is difficult. We discuss why the problem is computationally hard, and present an overview of a system to help users manage their tasks. We discuss the lessons learned from extensive simulation evaluations and a field test.

## Author Keywords

Activity coordination, team work, planning and scheduling

## ACM Classification Keywords

Office Automation, Time Management, Decision Support, Group and Organization Interface, Computer Supported Cooperative Work, Collaborative Computing, Distributed Artificial Intelligence, Intelligent Agents, Multi Agent Systems.

## INTRODUCTION

An important aspect of distributed team work is to determine which tasks to do, who should do them, when and where. Teams often have many options for organizing themselves and their work, with different options leading to different work plans with different durations, costs, risks, and quality of outcomes.

Our work focuses on systems to help a team of users manage their joint tasks so that they can efficiently adapt their

---

\*The work presented here is funded by the DARPA COORDINATORS Program under contract FA8750-05-C-0032. The U.S. Government is authorized to reproduce and distribute reports for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of any of the above organizations or any person connected with them. Approved for Public Release, Distribution Unlimited.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CSCW'08, November 8–12, 2008, San Diego, California, USA.

Copyright 2008 ACM 978-1-60558-007-4/08/11...\$5.00.

plans and schedules as work proceeds. Our system, called Criticality Sensitive Coordination (CSC) [5, 4], uses observable events and user input to monitor activity progress, and proposes options to repair plans and schedules to cope with delays, failures, new tasks, and new opportunities.

We envision a future where the costs and benefits of different collocation arrangements ranging from no collocation to radical collocation can be quantified and modeled. In such a world, teams would be faced with a large range of options for organizing their work. A system like CSC can help teams make best use of the opportunities offered by collaboration technology.

In this paper, we focus on the lessons learned after working on this problem for three years. This work was part of the DARPA COORDINATORS program, which funded three different approaches to the problem. DARPA ran competitive evaluations on simulated scenarios at the end of each year, and conducted a field exercise with real users at the end of the third year. CSC won the competitive evaluations in the first two years by a large margin (the last year results have not been made public yet) and also was the best system in the field evaluation.

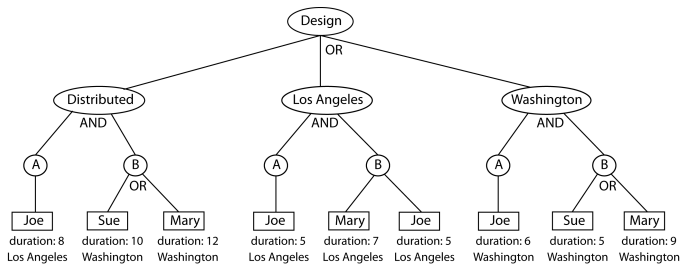
Next section introduces an example to illustrate the problems that we address in our work, and to connect our work to the goals of the workshop. Next, we describe the modeling framework, closing with a discussion of the lessons learned.

## EXAMPLE

We introduce a small example to illustrate different collocation options that may arise in a project. Consider a design project that involves the design of two components A and B. Joe, the lead designer is in Los Angeles, and Sue and Mary are in Washington. Joe, the lead designer can complete the design by himself. Sue and Mary can design component B. Figure 1 shows three collocation possibilities.

**Distributed:** Joe designs component A in Los Angeles taking 8 units of time. Either Sue or Mary can design component B in Washington taking either 10 and 12 units of time respectively. They use modern CSCW technology support to work on their designs.

**Los Angeles:** in this scenario, Mary could travel to Los Angeles and complete the design with Joe, or Joe could do the whole design himself. Sue is unable to travel so she cannot participate in the collocated design. The design



1: Collocation possibilities

times are reduced because the designers are collocated and can work more effectively.

**Washington:** in this scenario Joe travels to Washington. Because Joe is away from his office he cannot work as effectively as he would in Los Angeles, and consequently takes 6 units of time to complete the design of component A, instead of the 5 units he would take in Los Angeles. Either Sue or Mary can participate in the design of component B. The completion times are shorter than in the distributed case because the designers can work more effectively while collocated.

The collocation options seem more attractive given that the design times are reduced, and presumably the quality of the design would be best. However, while designers travel they cannot perform any useful activity and the cost must be added to the cost of the project. If this design was the only activity that these people are performing, it would be relatively easy to determine the best scenario.

In a real scenario, there would be more people and many more activities, there would be uncertainty in the duration of the activities given that problems may occur, priorities and availability could change, and new activities may arise. For example, a winter storm may abruptly increase the cost of travel. Managing the trade-offs becomes difficult, and should be done continuously as the situation changes.

## MODEL

A system to help a team of users manage their activities takes two types of inputs. The first is a continuously evolving model of the activities that the team needs to perform. Our modeling framework uses CTAEMS [1], a variant of TAEMS[3]. The main concepts in this framework are:

**Agents** represent the entities that can operate on the environment. In our case the agents represent people.

**Locations** represent places where work can take place. Agents can travel between locations.

**Methods** represent the primitive actions that agents can perform. Methods model the individual differences among users, and specify probability distributions for duration, cost, quality and outcome for each activity that a user can perform.

**Tasks** represent high level level procedures for accomplishing objectives. Tasks are decomposed into subtasks, and

the leaves of the task hierarchy are the methods. The task hierarchy is an AND-OR tree so it supports the representation of multiple options for performing tasks. The task model also includes accumulation functions that specify how quality, cost and duration flow up the tree (e.g., maximum, minimum, sum). The task model also includes support for modeling synchronized activities.

**Relationships** also referred to as non-local effects (NLEs) represent dependencies between nodes (tasks or methods) in the task hierarchy. The relationships supported are enables, disables, facilitates and hinders.

The models are dynamic. Users can define new tasks, methods and relationships as the situation evolves and can also adjust the deadlines, durations, qualities, etc. as appropriate. The task hierarchy in Figure 1 follows this model.

The second type of input is concerned with monitoring execution progress. In order to adjust plans and schedules, the system must be told when activities start and end, and where the agents are located. CSC provides screens to allow users to input this information, and also offers the ability to connect external sensing capabilities to automatically collect information about the status of activities. For example, in our disaster relief application the system uses GPS to detect the location of agents, and users are only required to tell the system when they complete an activity.

An important tool in the collaboration domain is a distributed activity manager that helps a distributed team manage multiple collaboration options (collaborative modeling of collaboration). The goal of CSC is to use these inputs to produce recommendations about who should do what, when and where. The algorithms that CSC uses to compute its recommendations are described in other publications [5, 4].

## MULTI-USER ACTIVITY COORDINATION IS HARD

Multi-user activity coordination superficially looks like traditional scheduling problems in that the problem could be viewed as one of computing the best schedule for each user. There are several aspects of the multi-user coordination problem that go well beyond the capabilities of traditional scheduling systems.

### *Uncertainty and Dynamism*

Human activities are inherently uncertain. The durations of tasks and travel are uncertain, and so are the outcomes. Activities may fail and may need to be redone. In addition, new activities crop up all the time, and priorities and resources change.

Traditional scheduling techniques, and traditional schedule models (e.g., MS Project) cannot cope with uncertainty. Uncertainty and dynamism lead to many possible futures, causing a combinatorial explosion that makes it impossible to solve optimally problems with only dozens of activities.

### *Distribution and Partial Observability*

An activity management system can be centralized or distributed. In the centralized approach, all models and user

inputs are sent to a central server that computes recommendations that it sends to users. The main problems with the centralized approach are scalability to many users and activities, and the ability to support intermittent connectivity.

In the distributed approach, each user is assigned an agent who manages the activities for a single user, and coordinates with other agents to compute solutions appropriate for the whole team. The difficulty in the distributed approach is that agents must make decisions based on partial information, making it difficult to compute globally good solutions.

#### *Bounded Computation and Communication*

The dynamic aspects of the problem imply that users want to see recommendations soon after they make changes. It is infeasible to compute for minutes because the system should be interactive. Bounded communication, especially if the system is running on small portable devices, further limits the algorithm options to share data among agents. We are faced with a large scale intractable problem where some level of solution must be given to users within seconds. It is not an option to run a massive computation off-line to compute the “best” schedule.

#### *Optimization*

Activity coordination is an optimization problem where we want to maximize the utility of the coordinated actions. Unfortunately, it is very difficult to define an appropriate global utility function that reflects the utility of all stake-holders. Our current work defines an objective function based on a measure of quality assigned to individual activities. The goal becomes to optimize the total quality accumulated at the root task subject to the constraints on the activities and resources available. Even this simple notion of utility yields intractable problems.

## **LESSONS LEARNED**

### **Avoid Failure**

When faced with a difficult optimization problem it is tempting to write heuristics to support some kind of heuristic search. The hope is that even if the heuristic is not accurate, the result steers the system towards good solutions.

The evaluation results obtained during the first and second year evaluations suggest that this is not a good approach. The problem is that the structure of activities produces a very complex non-linear objective function. The non-linearities arise from the and/or task decompositions, the enables and disables relationships and deadlines. The consequence is that a failure in one activity may cascade causing many other activities to fail so that certain small failures can cause significant disruptions. Our CSC system is specifically designed to avoid failure by using metrics to detect markers for many types of failure. We call such metrics criticality metrics. For example, the backbone metric is associated with every node in the task hierarchy, and it measures the fraction of a root task that will fail if that specific node fails. The backbone of a root task is 1.0 because if it fails, a root task fails. The backbone is propagated using message passing down

the AND/OR tree and backwards via enables links throughout the whole network. The backbone of the children of an AND node is equal to the backbone of the parent, but the backbone of the children of OR nodes decreases because all children must fail for the parent node to fail. In the end, it is possible to detect activities whose failure will have drastic consequences even though the node may belong to an agent who cannot see the whole network.

The CSC system uses several such criticality metrics, and uses a very simple algorithm for suggesting activities to the user. When a user becomes free, it uses a decision tree based on criticality metrics to select one of the activities that the user could start immediately. The system uses quality only as a tie-breaker when the criticality metrics are low. The CSC system does not do forward planning whatsoever.

The surprising result is that using this approach, CSC outscored the competing approaches [7, 6] by a large margin. In the second year evaluation, CSC obtained the best score on all but 12 of the 448 evaluation scenarios created by an independent evaluation team.

The lesson is that in computationally difficult problems a lot of mileage can be gained by simply avoiding failure. An additional benefit is that users are likely to lose trust on a system that sometimes leads to bad outcomes even if in general it leads to good outcomes.

### **Explainable Results**

Users need simple, concise justifications for the advice they get from systems. They have no time or patience to absorb complex arguments. Optimization systems often use complex algorithms or extensive search that find clever solutions involving long chains of reasoning. It is often hard to extract concise justifications from such long chains of reasoning.

Our criticality metric approach lends itself to concise explanations. Each criticality metric is designed to avoid particular failure modes, and it may suffice to tell users that a particular scheduling decision was done to avoid or alleviate a particular failure mode. For example, if the backbone value for an activity is high, the explanation is that it helps to keep options open for a specific high level task. Such a first order explanation may be enough for users, and if they request it, a second order explanation can be extracted to show the chain of backbone propagation where values are highest.

We don't yet have empirical evidence to validate this claim with end users. However, we have visualizations of our system that show the criticality metrics [2]. Using these visualizations we have been able to cut down the time to diagnose system behavior from hours to minutes. The visualizations are still too complex to be usable by end users, but they provide a first step towards intuitive explanations.

### **Mixed Initiative**

A mixed initiative system allows users to influence the solutions that the system produces. This is crucial in systems like this one where the objective function that the system

optimizes may not match the true utility function of the end users. The difficult question is to determine what level of input is appropriate for users to give.

Our experience indicates that a way to address this question is to divide decisions into strategic and tactical decisions. Users make the strategic decisions and the systems determines the tactical decisions to make implement the strategy.

CSC was used in a disaster recovery field exercise. In this scenario several towns suffered damage following a disaster. Surveys had to be done to assess damage. Injured had to be rescued and taken to medical facilities and utilities (power, gas and water) had to be restored. The rescue team had individuals with different skills. Services to clinics had to be restored before they could be used to assist injured, and it was necessary to travel to warehouses to fetch repair parts. Many repairs could be handled better (faster repair, no failure) if several repair people collocated and worked on the repair simultaneously. The amount of damage was such that it was not possible for users to rescue all injured and restore all utilities in the two hours allocated for the exercise. Points were allocated for rescuing injured and restoring services. Different points were allocated to different towns.

Three separate technologies were evaluated in this field exercise. There were three separate rescue teams, each using one of the technologies. One rescue team used tablet computers running the CSC software, and each member of the team was instructed to follow the advice of the computer. The second rescue team also used tablet computers running different software. The third rescue team was asked to coordinate using radios and no computer support.

Two separate scenarios were used. In the first scenario the team using the CSC technology came in first, the radio team was second and the other computer technology was third. In the second scenario the radio team was first, CSC was second and the other computer technology was third.

Even though reasonably accurate models of the problem were constructed, and the problem is essentially an optimization problem, and developers had been working on the system for three years, it was very difficult to do better than the radio team. We believe that the reason is that strategy plays a more important role than tactics. The radio team was able to fine-tune their strategy to each scenario, and furthermore, was able to fine-tune their strategy during execution. They made tactical mistakes due to the confusion that arises when 9 people are coordinating using radios, and each individual had poor knowledge of the global picture. We suspect they were able to mitigate some of these difficulties when individuals collocated at the sites and were able to more efficiently exchange information and formulate local plans.

The two computer systems use very different algorithms, but the biggest difference is that CSC allowed the users define a strategy. CSC allowed the users to split the rescue team into sub-teams and allowed them to assign a separate itinerary for each sub-team that specified the order for visiting the towns.

The system made the tactical decisions to determine what repairs to attempt at each site, who should do them, in what order, who should take injured to clinics, who should fetch repair parts, who should fix broken vehicles and remove road blocks. The reason that CSC lost in the second scenario is that it made one flawed tactical decision that resulted in almost all users sitting idle for about 10 minutes waiting for a pre-requisite activity to be completed.

We draw two lessons from this experience. First, avoiding failures is very important. Second, one can help the users most by focusing on the details and leaving the hard strategic decisions to the users. It is tempting to try to solve the whole problem, but taking care of the distracting details is a good first step towards helping them with hard problems.

### What-If

Given that strategic decisions are so important, it is useful for users to evaluate different strategies before committing to one. Interestingly, each of the field exercise scenarios was 2 hours long, but we spent over 4 hours brainstorming to develop the strategy for each scenario. Most of the discussion focused on predictions on how the strategy would unfold to predict potential flaws. Even though we had the ability to run low fidelity simulations of our strategies, we did not trust them because the stakes were too high. For example, the second scenario had a single warehouse, and the sites were divided into two clusters both of which were far from the warehouse. This was a crucial feature of this scenario, and it was below the level of fidelity of our simulations.

One lesson is that a high fidelity simulation is important to support a reliable what-if capability. An even more important lesson is that collocation is crucial when making the high stakes strategic decisions.

### REFERENCES

1. M. Boddy, B. Horling, J. Phelps, R. P. Goldman, R. Vincent, A. C. Long, B. Kohout, and R. Maheswaran. CTAEMS language specification: Version 2.04, 2007.
2. J. Jin, R. Sanchez, R. T. Maheswaran, and P. A. Szekely. Vizscript: on the creation of efficient visualizations for understanding complex multi-agent systems. In *Intelligent User Interfaces*, pages 40–49, 2008.
3. V. Lesser, K. Decker, T. Wagner, N. Carver, A. Garvey, B. Horling, D. Neiman, R. Podorozhny, M. N. Prasad, A. Raja, R. Vincent, P. Xuan, and X. Q. Zhang. Evolution of the GPGP/TAEMS domain-independent coordination framework. *Autonomous Agents and Multi-Agent Systems*, 9(1-2):87–143, 2004.
4. R. Maheswaran and P. Szekely. Criticality metrics for distributed plan and schedule management. In *Proceedings of the 18th International Conference on Automated Planning and Scheduling*, Sydney, Australia, September 2008.
5. R. T. Maheswaran, P. Szekely, M. Becker, S. Fitzpatrick, G. Gati, J. Jin, R. Neches, N. Noori, C. Rogers, R. Sanchez, K. Smyth, and C. VanBuskirk. Predictability & criticality metrics for coordination in complex environments. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*, Estoril, Portugal, May 2008.
6. D. J. Musliner, E. H. Durfee, J. Wu, D. A. Dolgov, R. P. Goldman, and M. S. Boddy. Coordinated plan management

using multiagent MDPs. In *Proceedings of the 2006 AAAI Spring Symposium on Distributed Plan and Schedule Management*, March 2006.

7. S. Smith, A. T. Gallagher, T. L. Zimmerman, L. Barbulescu, and Z. Rubinstein. Distributed management of flexible times schedules. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS 2007)*, Honolulu, HI, May 2007.