NAVCo: Negotiation-based Adaptive View Coordination

Prasanta Bose
George Mason University
bose@isse.gmu.edu

Mark G. Matthews
The MITRE Corporation
mmatthew@mitre.org

Abstract

In mission critical applications of distributed information systems, autonomous information resources are coordinated to meet the information demands of client specific decision-support views. The current approach to view coordination relies on design-time trade-offs to select a static view coordination policy from a set of available policies. This approach is not robust and does not respond well in a dynamic environment with shared infrastructure, and dynamically changing missions, priorities, preferences, and constraints. This paper introduces, NAVCo, a negotiation-based adaptive view coordination approach that allows view coordination policies to be dynamically negotiated and adapted at run-time in response to dynamically changing conditions.

1 Introduction

In distributed decision-support database systems, autonomous information resources are coordinated to meet the information demands of client specific views. A major challenge is handling dynamic changes in quality of service (QoS) properties and constraints of the clients, resources, and shared infrastructure. Current view coordination approaches are static in nature and cannot be dynamically changed to meet changing demands.

Consider the following scenario where a decision-support view for inventory management is based on multiple autonomous information resources within a supply-chain. Customer order information from customer sites, product assembly information from manufacturer sites, and parts inventories from parts supplier sites are configured to support an order-fulfillment view used by inventory managers of the suppliers and consumers. As orders, product assembly requirements, and parts inventories constantly change, changes in the view must be coordinated to achieve consistency and to support management decisions.

There are multiple view change coordination policies available to maintain distributed decision-support database views. These policies provide tradeoffs between consistency, communications costs, and processing costs. Currently, the view coordination policy is selected from a set of available policies at design-time and can not be dynamically adapted during run-time. The current approach is inflexible and fails to support a dynamic environment that relies on a shared infrastructure with inherently conflicting requirements. Suppose the view is maintained through a high-cost complete consistency policy while several users are simultaneously executing resource intensive queries. The queries are competing with the executing coordination policy for resources. Most likely, both the queries and the view maintenance will suffer from poor performance. Short of shutting down, reconfiguring, and restarting the system, current approaches have no way of responding to dynamically changing missions, priorities, preferences, and constraints.

This paper introduces a negotiation-based adaptive view coordination (NAVCo) approach that allows view coordination policies to be dynamically adapted to support a dynamic run-time environment. NAVCo incorporates the following key ideas: a) a negotiation model for adapting view maintenance policies to meet changes in QoS needs and constraints; b) a dynamic software architecture of the collaborating information resources supporting the client task of maintaining a specific view; and c) adaptive mechanisms in the architecture that realize negotiated changes in the coordination policies for view maintenance.

2 Multi-Resource View Coordination

View coordination objects (VCOs) are algorithmic objects that correspond to different policies for view maintenance. These view maintenance algorithms focus on maintaining the materialized view at the client in the presence of concurrent updates to the data resources.

The performance of VCOs can be compared based on the communications and processing costs required to maintain a certain level of consistency. Communications costs can be measured with respect to the number and size of messages required per update. Processing costs can be measured with respect to the processing burden that the algorithm places on both the client and the data sources.

Table 1 compares the communications and query processing cost of four existing VCOs. The Strobe and Nested SWEEP algorithms provide strong consistency while the C-Strobe and SWEEP algorithms provide complete consistency. A complete description of these

0-7695-0415-9/99 $10.00 © 1999 IEEE

algorithms can be found in [1, 8]. The cost of the algorithms is dependent on the number of data sources, n. The costs of the C-Strobe and Nested SWEEP algorithms are highly dependent on a workload characterization factor, a where $0 \leq a \leq 1$, which reflects the rate of updates received. If updates arrive infrequently $a=0$ and if updates arrive continuously $a=1$. The client processing cost of a delete update in the Strobe and C-Strobe algorithms is highly dependent on the number of pending updates, p.

Table 1 – VCO Cost Comparison

Algorithm	Update Type	Comm Cost	Client Cost	Server Cost
Strobe	delete	1	1+p	0
	insert	2n-1	(n-1)+1	1
C-Strobe	delete	1	1+p	0
	insert	2(n-1)+ 2a(n-1)!+1	(n-1)+ a(n-1)!+1	1+a(n-2)!
SWEEP	delete/insert	2n-1	(n-1)+1	1
Nested SWEEP	delete/insert	2(1-a)(n-1)+1	(1-a)(n-1)+1	(1-a)

As illustrated in table 1, the cost of the algorithms is highly sensitive to the volume and types of updates. To illustrate, consider a supply-chain view maintenance application with four data resources, one client view, and the following dynamic workload:

Period 1 -high volume, high insert workload, 100 inserts and 0 deletes over X seconds

Period 2 - low volume, balanced workload, 50 inserts and 50 deletes over 3X seconds

Period 3 - medium volume, high delete workload, 0 inserts and 100 deletes over 2X seconds

The cost of each algorithm over these periods can be calculated using the formulas in table 1. The value of the parameter p is assumed to be 0 for low traffic, 10 for medium traffic, and 100 for high traffic. The value of the parameter a is assumed to be 0 for low traffic, 1/3 for medium traffic, and 1 for high traffic. The cost of the four algorithms over Periods 1-3 is illustrated in table 2.

Currently a single algorithm is selected at design time and can not be changed without shutting down and reconfiguring the system. Design-time tradeoffs must be made with respect to consistency versus client, server, and communications costs. The design-time decision can have a profound effect on the processing and communications requirements to support the view. If, however, the algorithm can be dynamically changed at run-time, these tradeoffs can be made continuously as preferences and constraints change. As illustrated in the two examples at the bottom of table 2, the ability to dynamically switch algorithms can result in significant cost savings and improved performance in a constrained environment.

Example 1 shows that communications cost can be minimized by initially implementing the Nested SWEEP algorithm and then dynamically switching to the Strobe algorithm between periods 1 and 2. This results in savings of 450 messages over a static implementation of the Strobe algorithm. This frees up valuable shared communications resources for more critical applications.

Example 2 shows that client processing cost can be minimized by implementing the Nested SWEEP algorithm during periods 1 and 3, and the Strobe algorithm during period 2. This results in savings of over 1000 queries over a static implementation of the Strobe algorithm. This frees up valuable resources for processing-intensive analysis queries and results in a significant performance improvement for analysis users.

Table 2 –Cost in Supply-Chain Scenario

Algorithm	Comm Cost	Client Cost	Server Cost
Strobe	1200	1750	150
C-Strobe	2400	2350	350
SWEEP	2100	1200	300
Nested SWEEP	1250	775	158
Example 1	750	1525	75
Example 2	950	625	108

3 The NAVCo Approach

The NAVCo approach to adapting view coordination policies is based on negotiation reasoning between the client and resource objects. The approach introduces negotiation reasoning models and adaptive policy reconfiguration mechanisms to the existing view coordination application architecture. The NAVCo architecture, shown in figure 1 as a UML class diagram, introduces a negotiation layer to perform dynamic negotiation-based selection of coordination policies. Additional software mechanisms are introduced to bring about the dynamic switching of the coordination objects.

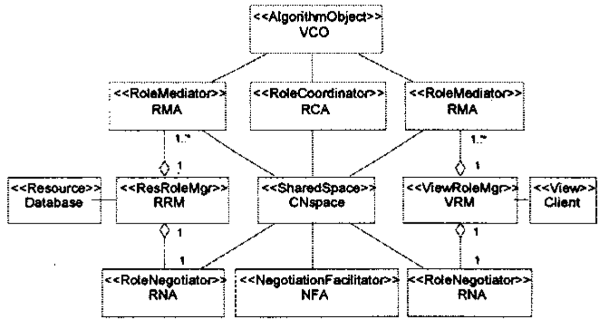

Figure 1 – The NAVCo Architecture

The NAVCo architecture contains models and reasoning support for model-based coordination negotiation via role negotiation agents (RNA), and negotiation facilitation agents (NFA) that communicate via a shared coordination negotiation space (CNspace). The architecture also contains models and support for switching based on automated negotiated switching decisions. The application and negotiation layers are integrated via the shared data space whereby the negotiation facilitation agent communicates with the team level coordination agent via the CNspace. The role coordination agent (RCA) and role mediator agents (RMA) aid in coordinating the dynamic switching.

3.1 Negotiation Reasoning

The model for negotiation reasoning is an extension of the WinWin [2] model used in requirements negotiation. The participating agents collaboratively and asynchronously explore the WinWin decision space that is represented by four main conceptual artifacts: 1) WinCondition - capturing the preferences and constraints of a participant. 2) Issue - capturing a conflict between WinConditions or associated risks and uncertainties. 3) Option - capturing a decision choice for resolving an issue. 4) Agreement - capturing the agreed upon set of conditions which satisfy stakeholder WinConditions and/or agreed options for resolving issues.

NAVCo considers three types of negotiation reasoning schemes that extend WinWin to incorporate a reactive negotiation model. The first method, used during the initial establishment of the task and negotiation of the initial policy, takes a task-driven approach and is triggered when a new WinCondition of a client is submitted. In this scheme, the client initiates the task through the submission of a WinCondition containing the task parameters and any client preferences and constraints.

The second method, used for run-time dynamic renegotiation of policies, is conflict driven and is triggered by changes in preferences and constraints. In this scheme any team participant may submit a revised WinCondition based on changing component preferences and constraints. The protocol for this method is contained in figure 2.

1. Resource or client RNA posts a change in preference, priority, or constraint as a WinCondition revision
2. NFA analyzes the revision in the context of agreed WinConditions to generate issues resulting from pairwise conflicting interaction
3. NFA generates options for issues
4. If there is no change in option then NFA marks the issue as resolved and process STOPs
   Else Resource and client RNAs evaluate the option and post option evaluations
5. If evaluation accepts option and all issues resolved then STOP
   Else RNA revises WinCondition and go to step 2

Figure 2 –Conflict Driven Negotiation Protocol

The third method is priority-driven and is used when an acceptable policy can not be negotiated among all team participants in a predetermined amount of time. In this scheme, team participants are assigned a priority based on inputs from the task owner. The option with the highest overall utility, based on global and team member priorities, is selected.

Given the above reasoning methods three major technical challenges are generating issues, generating options, and evaluating options. The NAVCo approach exploits the view coordination problem domain to address these challenges as follows.

Issue Generation: Given one or more WinConditions, issue generation involves formulating a query to identify VCO specifications that satisfy the WinConditions. Here the issue is formalized as a query object.

Option Generation: Given the formulation of the issue, option generation involves evaluation of the query to retrieve plausible VCO specifications and refinements.

Option Evaluation: Given options, option evaluation involves checking for consistency of an option against a database of committed WinConditions.

3.2 Models for Negotiated VCO Selection

In order to support the reasoning approach outlined above, NAVCo requires a) declarative models of preferences and constraints at the clients and resources as a database of facts, and b) rules for issue generation, option generation and evaluation. We briefly describe below the RNA and NFA data models, and several of the rules that have been formulated and prototyped.

The RNA data model articulates the WinCondition as consisting of two parts: a) task part - articulates the role to be played, prioritization of tasks, task preferences, and update volume and distribution submitted to the team in support of the task. b) QoS constraint part - articulates the constraints imposed on the task. The QoS schema specifies the component workload to support the task and the component QoS constraints based on the status of component resources captured as QoS metrics. The data model also specifies global integrity constraints.

The NFA data model captures VCO specifications and their associated costs. The data model also contains models of the WinConditions, issues and options that get posted or generated by the NFA. Some of the important data elements are a) identification, characteristics, and costs of available coordination policies, b) task-specific meta-data, and c) overall team-level workload characterization, preferences, and constraints.

The rules for issue and option generation, and option evaluation are modeled as database trigger rules that analyze WinCondition updates and create associated issues, options, and option evaluations. The NFA issue generation trigger rule is triggered by an update to the NFA WinCondition relation and creates an issue, whose semantics are that of a query assertion to select a VCO meeting relevant preferences and constraints imposed by task specific WinConditions. The NFA option generation trigger rule is triggered by an issue entry in the NFA issue relation and executes the query assertion to identify options to insert into the NFA options relation.

3.3 Mechanisms for Dynamic VCO Switching

NAVCo contains adaptive software mechanisms to allow VCOs to be dynamically switched in response to the negotiated selection of a VCO. The NFA propagates negotiation results to the application view by writing a dynamic switching plan (DSP) object into the CNspace. The DSP identifies the VCO that has been negotiated and includes detailed executable instructions for dynamically switching between VCOs. Each RMA and NFA reads the DSP from the CNspace and executes the instructions that pertain to them. The basic approach is to create a new VCO of type identified in the DSP, transfer the workload to the negotiated VCO, and finally to destroy the outdated VCO. Dynamic bindings between RMAs and VCOs are handled through the use of the Jini Lookup Service.

4 Prototype

RMA, RNA, NFA, and RCA prototypes have been developed. Each prototype agent consists of a Java application and a Microsoft Access database. All agent-to-agent coordination is accomplished through the CNspace, which is implemented as a JavaSpace. WinConditions, options, dynamic switching plans and other objects are written as entries into the CNspace. The CNspace notify and read methods are utilized to route the entries to the appropriate agents. The prototype agents currently use input and output text files to simulate interactions with clients and resources. Negotiated reasoning and dynamic switching experiments have been conducted. Initial results show that the NAVCo reactive reasoning methods work well within the JavaSpaces environment.

5 Related Work

There has been a significant amount of work conducted in the area of view maintenance resulting in a spectrum of solutions ranging from a fully virtual approach where no data is materialized at one extreme to a fully replicated approach where full base relations are copied at the other extreme. The Strobe [8] and SWEEP algorithms [1] are a hybrid of these two extremes and are designed to provide incremental view maintenance over multiple, distributed resources.

NAVCo builds on research using the intelligent agent approach to automated negotiation. The agent approach focuses on computational agents that negotiate to resolve conflict [3], to distribute tasks [5, 7], to share resources [9], and to change goals so as to optimize multi-attribute utility functions [6]. The WinWin [2] model used in NAVCo considers negotiation between multiple agents driven by both global and independent local utility functions.

NAVCo is similar in spirit to work on architecture-based run-time evolution [4]. NAVCo differs from [4] in terms of the nature of automation. While [4] focuses on providing a support environment where the analysis for dynamic change and consequent adaptation can be performed, NAVCo is motivated by automated switching based on negotiation reasoning.

6 Summary

This paper presents a negotiation-based adaptive view coordination (NAVCo) approach that allows view coordination policies to be dynamically coordinated and adapted to support a dynamic run-time environment. This approach supports dynamically changing missions, priorities, preferences, and constraints. The approach supports the optimal allocation of shared infrastructure among competing applications and tasks. The paper presents the key ideas and models developed and prototyped in our initial experiments with the approach.

7 References

[1] D. Agrawal, A. El Abbadi, A Singh, and T. Yurek, "Efficient View Maintenance at Data Warehouses", *Proceedings of ACM SIGMOD '97*, 1997, pp. 417-427.

[2] B. Boehm, P. Bose, E. Horowitz and M..J. Lee, "Software Requirements Negotiation and Renegotiation Aids: A Theory-W Based Spiral Approach", *Proceedings of 17th ICSE*, 1995.

[3] E.H. Durfee, V.R. Lesser, and D.D. Corkill, "Cooperation Through Communication in a Distributed Problem Solving Network", M.N. Huhns ed., *Distributed Artificial Intelligence*, Academic Press/Morgan Kaufmann, 1989, pp. 29-58.

[4] P. Oreizy, N. Medvidovic, and R.N. Taylor, "Architecture-based Runtime Evolution", *Proceedings of ICSE 1998*.

[5] R.G. Smith, "The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver", *IEEE Trans. on Computers*, Vol. 29, 1980, pp. 1104-1113.

[6] K.P. Sycara, "Resolving Goal Conflicts Via Negotiation", *Proceedings of AAAI-88*, 1988, pp. 245-250.

[7] M.P. Wellman, "A General Equilibrium Approach to Distributed Transportation Planning", *Proceedings of AAAI-92*, San Jose, CA 1992.

[8] Y. Zhuge, H. Garcia-Molina, and J. Wiener, "The Strobe Algorithms for Multi-Source Warehouse Consistency", *Proceedings of International Conference on Parallel and Distributed Information Systems*, December 1996.

[9] G. Zlotkin, and J.S. Rosenschein, "Mechanism Design for Automated Negotiation and its Application to Task Oriented Domains", *Artificial Intelligence*, Vol. 86, 1996, pp. 195-244.