A Framework for Collaborative Updates in Selective Data Replication Communities

November 8, 2007

Simon Fox

smf46@student.canterbury.ac.nz

Department of Computer Science and Software Engineering

University of Canterbury, Christchurch, New Zealand

Supervisor: Dr. Richard Pascoe

richard.pascoe@canterbury.ac.nz

Abstract

Collaborative replication updates are an attractive property of Selective Data Replication in which data consumers cooperate to update their data set replications. Data consumer communities are implemented as groups of intelligent software agents who make decisions about when updates should occur. The software agent paradigm is suitable for achieving collaboration between individual agents, however some structured collaboration model must be followed. The theory of Cooperative Problem Solving (CPS) [19] describes a theoretical model for achieving collaboration between a group of software agents. We present a framework implemented using the OPAL agent platform, and derived from the CPS theory, for achieving collaborative replication updates within a community of data consumers.

Contents

1 Introduction

Selective Data Replication (SDR) is a technique used to reduce the costs related to updating a replication of a data set. Updates are required of a replicated data set to reflect changes which occur at the source of that data set. The reduction in costs is achieved by making intelligent decisions about when and how an update should be performed, rather than blindly scheduling updates as is the case in traditional data replication.

Collaborative replication updates are an attractive property of Selective Data Replication in which data consumers cooperate to update their data set replications. Data consumer communities are implemented as groups of intelligent software agents who make decisions about when updates should occur. The software agent paradigm has been identified as suitable for achieving collaboration between individual agents, and there exist a number of collaboration theories that have been developed with respect to agents.

As is the case in collaborative action between a group of people, agents must be able to communicate and negotiate the terms of collaboration, and make decisions about when collaboration is required and how it should be performed. These requirements have the consequence that the agents which participate in collaborative updates must have a level of intelligence. The OPAL agent platform (described in Secion 2.3) provides the infrastructure required to create intelligent software agents, these capabilities will be required in this project.

1.1 Goals and Objectives

The goal of this project is to develop a framework using the OPAL agent platform that supports the execution of collaborative updates within Selective Data Replication communities. The creation of this framework will be guided by the literature that has been developed in the area of collaboration theory. Such theories describe collaboration between groups of agents at an abstract level that is domain independent. Overcoming this abstraction and applying the concepts derived in these theories to the domain of collaborative updates in SDR will be one of the major challenges.

The collaborative theories will be examined in detail, and the theory most suited to creating a framework for collaborative updates will be chosen as a vehicle for guiding framework development. Preliminary work [5] has identified three collaborative theories that have been well received in the research community, and may be suitable for development of the framework. These theories are described in Section 2.4.

At the current time, there exists no other work in the area of collaborative updates in selective data replication. In order to develop a framework for achieving them, what it means to update collaboratively must be defined. Section 3 describes a set of scenarios in which a group of agents representing emergency service organisations wish to collaborate in order to update replications of roading data sets that are used to determine emergency routes that are unaffected by roadworks. These scenarios will both guide the formal derivation of a collaborative framework, and act as a goal for which a successful framework implementation should achieve.

2 Related Work

2.1 Selective Data Replication Communities

In a selective data replication community the replication of data consists of two roles, these roles can be viewed like a master-slave configuration, where the master provides a data set which is replicated by the slave. The roles are described for the remainder of this paper using the following notation taken from [13]:

The data provider Data provider, P_n, is an organisation which supplies access to a data set λ

The data consumer Data consumer, C_m, is an organisation which replicates data set λ

A data producer allows consumers who replicate the data set he provides, to subscribe to change notifications that will inform the consumer of changes occurring in the data set [13]. The content of these notifications relies heavily on the context in which the data replication is being performed. In Section 3 three scenarios of collaborative updates are derived in the context of a geographical information system, the content of change notifications is therefore relevant to the domain of GIS data sets.

A Selective Update Policy is the policy a data consumer uses to make decisions about when to schedule an update. The policy defines a set of rules which specify conditions under which an update should be scheduled. It is within this policy that the conditions under which a data consumer will attempt to achieve a collaborative replication update are defined.

2.2 Beliefs, Desires and Intentions Architecture

Selective data replication requires participating data consumers to make decisions based on the employed selective update policy, and the current state of the replicated data source. Decision making based on knowledge and analysis of the current state of the world constitute rational behaviour [1]. The Beliefs, Desires and Intentions Architecture (BDI), is popular in the design and development of rational agents [20], and preliminary work on update policies concluded that it is a suitable architecture for creating data consumer agents [5].

Beliefs represent the information state of the agent [6], and provide a set of facts that the agent can reason about. The information state is used to represent the agents knowledge of the current world model. Desires represent the motivational state of the agent [6], a set of goals that the agent wishes to achieve. Intentions are plans that an agent has adopted as a result of an unsatisfied desire [6], and the agent believes that execution of the plan will alter the world state in such a way that the desire becomes satisfied.

2.3 The OPAL Agent Platform

OPAL is the Otago Agent Platform, it provides a framework for creating Software agents which conforms to the specifications set out by the Foundation for Intelligent Physical Agents (FIPA) [4]. The FIPA specification defines a number of services that must be provided by a compliant agent platform, of interest to this project are the Agent Directory Service and the Message Transport Service described in Sections 2.3.2 and 2.3.3 respectively. OPAL also provides three reasoning engines to allow for the creation of intelligent agents, the ROK reasoning engine has been identified as suitable to SDR in preliminary work which is described briefly in Section 2.3.1.

2.3.1 Rule-Driven Object-Oriented Knowledge Base System

ROK, Rule-driven Object-oriented Knowledge base system, is a reasoning engine derived from JEOPS [17, 3] that is integrated in to the OPAL platform to allow for the creation of intelligent software agents. ROK provides Java with a mechanism for inserting declarative rules in to an application. ROK production rules are described by the condition-action pattern, if a certain condition holds then its associated actions should be performed.

A ROK rule has three parts: declarations, declare the objects that are used in the condition and action parts of the rule; conditions, use function calls on the declared objects and standard Java operators to compare values and test some condition against the current state of the objects; and actions, which declare what should be performed if the conditions were satisfied. Any Java object can be used in a ROK rule, as long as the object appears in the declarations of the rule.

Preliminary work determined that the ROK reasoning engine can be used to achieve the BDI agent architecture required to create data consumer agents [5]. In short, the declarations of the ROK rule define objects that must exist in the beliefs of the agent for the rule to be evaluated. The conditions of each rule define a goal or desire the agent wishes to be true of its beliefs. The actions of the rule are taken on as intentions of the agent if the conditions are not satisfied. The beliefs of the agent are stored within an instance of the KnowledgeBase class which provides a database like store for objects. The ROK reasoning engine can be setup to monitor the knowledge base using an implementation of the Observer pattern, allowing evaluation of the agents desires whenever a change occurs in his beliefs.

2.3.2 Agent Directory Service

The Agent Directory Service (ADS) is defined by FIPA and an implementation is provided by the OPAL platform. The ADS provides agents registered with the platform a yellow pages style look up. Any agent who wishes to broadcast a service he provides or the interests he has, can register a description with the ADS. Any agent who wishes to determine a provider of a service of a specific type or an agent with interests common with his own, can request the ADS to provide a set of agent descriptions based on a search template.

2.3.3 Message Transport Service

The Message Transport Service (MTS) is defined by FIPA and an implementation is provided by the OPAL platform. The specification defines the MTS as "a message transport service which handles the delivery and reception of messages between agents". Once an agent puts a message on to the service, delivery of that message is guaranteed to occur. The MTS removes the need for individual agents to manage issues related with message delivery, once passed to the service the agent can continue with other tasks and an agent can be sure that he will be notified if a message addressed to him is put on the service (asynchronous communication).

Messages

The OPAL implementation of messages is relatively simplistic and results in consequences for sending detailed or complex information structures between agents. A message in OPAL consists of the required sender and receiver addressing information, a message type (for example request or inform), and a set of key-value tuples of type string. The result of this implementation is that detailed information structures that need to be sent between two agents must be represented in a string form and appended to the message as a key-value tuple. Conversion of such structures may require some amount of processing or parsing to allow transmission via the MTS.

2.3.4 Micro-agents and Roles

The Micro-agent architecture is a unique feature implemented by OPAL. It provides a way of creating agents at a finer granularity than that used in the standard software agent architecture. A software agent which exists on an agent platform is considered to be an aggregation of micro-agents in OPAL. Each micro-agent exists to provide a single piece of functionality, “streamlined agents that can be used for conventional, system-level programming tasks“ [15]. The result of combining a set of micro-agents, is a fully functional, software agent that takes advantage of the notions of Abstraction, Decomposition, and Organisation [15].

A futher feature OPAL provides to micro-agents is the concept of an agent Role. A mirco-agent can play any number of roles within the aggregate agent. In simple terms, roles are defined by the user by extending the Role interface provided by OPAL, they define a set of behaviours or responsibilities that an agent can perform. When micro agents are loaded through the platform, the Roles they play can be specified. The platform can then provide services such as micro-agent look up based on the required Role.

2.4 High Level Theories of Agent Collaboration

The domain of multi-agent collaboration (MAC) has emerged from the research in artificial intelligence in distributed problem solving [19, 18]. This section provides an overview of three high level MAC theories (Joint Intentions [2], Shared Plans [11], Planned Team Activity [9]) which have provided an influential contribution to the area [18]. It is important to note that these are not the only MAC theories that have been developed, the existent theories are numerous due to the fact that there is no single definition of multi-agent collaboration that can be applied universally [7, 18]. The three theories covered here have been chosen due to the recognition they have received and the influence they have had on the emergence of the Cooperative Problem Solving Theory (described in Section 2.5) which has been used in the development of the proposed framework.

Each of the theories described here have groundings in Psychology and Cognitive Science, they describe high level, formal models using first-order logic which attempt to identify core requirements of a collaborative agent system. At this level, agents are simply actors in the system and represent a super set of any entity that may be capable of, or require collaborative ability. This broad target introduces a level of abstraction which must be overcome in order to provide an implementation suitable for use with software agents who exist in a particular domain.

2.4.1 Joint Intentions

The theory of Joint Intentions attempts to answer the question “What is involved when a group of agents decide to do something together?” [2]. Collaborative action is not considered to be simply the act of simultaneous individual actions by members of a group, but the act of performing an individual action which moves a group closer to a shared goal, with the belief that all other group members know that action is being performed. The Joint Intentions theory focuses on the mental state of agents within the group while those actions are performed, and the responsibilities of each of those agents toward group members.

As the name suggests, intentions are a key component of the Joint Intentions theory. The idea of an intention comes from the Belief, Desire, Intention (BDI) agent architecture [6, 5], which defines an intention as the commitment of an agent while in a certain mental state [18] to perform some action in order to achieve a desired goal. The Joint Intentions theory extends the idea of a single agent intention as defined by the BDI architecture, to a joint intention which is held between the agents of a group.

The authors (Cohen and Levesque) provide a formal definition of their theory using a temporal logic augmented by propositional terms [18], this definition is complex and such logic is not required for the purposes of this project, therefore it is omitted here. The following informal definition attempts to capture the important aspects and is followed by an explanation.

Mutual belief A belief that is shared between the members of a group. If a group of agents hold a mutual belief, each agent in that group both believes the piece of knowledge being represented and believes that the members of his group believe that piece of knowledge.

Mutual goal A goal that is shared between the members of a group. Each agent in the group believes that his peers all have the desire to bring about the goal.

Weak Mutual goal A goal held by an individual agent, to make his private beliefs known to the group. Joint Intentions requires that the group be in a “shared mental state“ [2], this state is achieved through the use of the weak mutual goal. If an agent adopts a private belief, the weak mutual goal ensures he makes that belief known to the group.

Joint Persistent goal A commitment which binds a group of agents to achieving some shared goal. The joint persistent goal is created from: a mutual belief that the goal of interest is not yet achieved; a mutual goal to achieve the goal of interest; a commitment to a weak mutual goal, ensuring that any private belief adopted by any agent in the group will be made known to the group i.e. will become a mutual belief.

The Joint Intentions theory does not specify any technique a group should employ to determine how a joint persistent goal will be achieved. Any group member can perform any action which may move the group closer to achieving the goal. Because the weak mutual goal is in place, the rest of the group will be informed of that agents intention to perform, and the result of, that action. This leads to an important insight highlighted by the authors, if an agent comes to a private belief that the goal has been achieved, is unachievable, or is irrelevant, the existence of the weak mutual goal ensures that this belief will be propagated to all members of the group [2].

The shared mental state is the driving factor of collaborative action in the Joint Intentions theory. If a group of agents all have the same beliefs about the state of their environment, and share a goal which is not yet achieved in that environment, they will all take steps towards achieving that goal and communicate any changes to the environment which may have an effect on the goal. Obtaining such a shared mental state and acting upon it introduces a number of significant overheads: a large knowledge base maintaining the state of the environment as experienced by the agent as well as by his peers; communication of changed or new beliefs whenever a change in the environment effects the current mental state of an agent; a large library of actions that can be performed, and the relevant situations under which they can be used to achieve specific goals.

2.4.2 Shared Plans

The Shared Plans model of collaboration takes a more *action focused* stance on the problem of agent collaboration. The aim is to model how the intentions (defined in Section 2.4.1 as the commitment of an agent to perform some action in order to achieve a desired goal) of different agents contribute to achieving their overall goal [12]. Collaborative action is guided by a plan of action or recipe which describes the subactions that need to be performed, the agent that performs each subaction, and the time interval over which that subaction will be executed. In contrast to the Joint Intentions theory, Shared Plans was developed for recipes in which each of the (sub)actions is performed by one of two agents i.e. the group of agents is of size two.

Like the Joint Intentions theory, the authors of Shared Plans use a first-order logic to define what it is to have a Shared Plan [11, 12]. Again, this formal specification will be omitted for its complexity, and the following informal explanation is used to describe the important aspects.

Mutual belief A belief that is shared between the two agents developing the shared plan. The Shared Plan theory adds the restriction that the mutual belief exists over a specified time interval.

Recipe for Action A recipe for action, is a decomposition of an action that may achieve some overall goal, in to act-types or subactions that can each be performed by an individual agent and the relationships between those act-types.

Execute Execute is a predicate which holds if an agent has the ability to perform a specific act-type or subaction of the shared plan.

Intends Intends is a predicate which holds between an agent, an act-type or subaction of the shared plan, and a time interval over which the agent will execute the associated subaction.

Shared Plan The plan which is shared by the two agents, describing a set of act-types, to accomplish a shared goal achieved by execution of a composite action which is the sum of the defined act-types.

Using the these definitions, a shared plan is constructed from the following: a mutual belief that at least one of the agents can *execute* each act-type defined by the recipe; a mutual belief that the chosen *recipe for action* does decompose the chosen action for achieving the overall goal; a mutual belief that for each act-type of the chosen recipe, one of the agents is committed to its execution at a specified time through an *intends* belief and that performing the subaction defined by that act-type a contribution will be made to the chosen action; private *intends* beliefs for each of the act-types defined by the recipe that the agent in question has chosen to perform.

The major difference of the Shared Plans theory to Joint Intentions, is the lack of the requirement for a shared mental state. As the plan is created each agent will become aware of who will perform each subaction of the plan through the mutual belief for each act-type. However at no time does either agent have a responsibility to make his private beliefs known, and as a consequence, if an agent comes to privately believe that the goal in quesion is either achieved, unachievable or irrelevant, he has no responsibility to inform his peer. The result is that there is no clear explanation as to what will happen to the joint action under such a situation, an agent may have a private belief that he should inform his peer of this situation, however the theory provides his peer has no guarantee that he will be informed.

2.4.3 Planned Team Activity

In contrast to the previous two collaboration theories in which a team is formed immediately and completely through either a joint commitment or a shared plan, the Planned Team Activity theory allows for ”expressions of interest“ [18]. Planned Team Activity describes two methods of team formation, both achieved through a team leader and the communication that occurs with potential team members. Further, this theory requires that plans for collaborative action be known by all potential collaborators in advance to actually joining a team.

Team formation is initiated by an agent who wishes to achieve some goal, but is unable to do so alone [18, 9]. This agent then proceeds in attempting to create a team who together, is capable of achieving the goal. Because all potential collaborators are aware of the plans for collaborative action, and the tasks or roles that must be filled within those plans, the team leader has only the responsibility to decide on the plan which will be used, and the agents who will perform the roles of that plan.

Two strategies for team formation are described by the authors of Planned Team Activity: commit-and-cancel; and agree-and-execute. A short explanation of each follows.

Commit-and-Cancel The team leader sends a request to each potential participant. The request includes the joint goal, the plan that will be executed to achieve that goal, and the role that the participant receiving the request will play in that plan. Participants must then decide if they accept the terms that the team leader has put forward and if so, reply as ”committed“. If the team leader receives a ”committed“ reply from all the participants, the team has been formed and execution of the plan begins immediately [18]. If any participant does not commit, or does not reply within a specified time-out, the team leader inform all committed participants using a ”cancel“ message.

Agree-and-Execute The team leader sends an ”agree“ request to each participant and waits for a reply. If all participants reply in agreement, each is sent a request to execute the plan. Because the participant has already agreed to the collaborative action, the execute request specifies the role of the plan that the participant should play [9], a joint goal is also adopted at this time to maintain the agents responsibility to the team.

Like Joint Intentions, if an agent becomes unable to perform his assigned role as part of the collaborative action, he has a responsibility to inform the other members of the team. However, Planned Team Activity requires that the agent who caused the failure coordinate a response, in effect taking on a team leader role and informing the members of the action they should take (this could simply be to disband if no other options exist).

2.5 The Cooperative Problem Solving Theory of Collaboration

The Cooperative Problem Solving Theory (CPS) [19] defines a four stage model of collaboration. Each stage defines a set of tasks required to achieve collaborative action between a group of agents. The model considers collaboration from a single agent realising a potential, through to a group of agents executing a collaborative action to achieve a common goal. The four stages of the mode are: Recognition; Team Formation; Plan Formation; Team Action.

2.5.1 Recognition

The Recognition stage of the CPS model begins when an agent recognises the potential for cooperative action with respect to some goal [19, 8]. Recognition may occur either if an agent is unable to achieve a desired goal in isolation, or if an agent is able to achieve the goal, however does not want to consume the required resources and therefore decides to solicit assistance.

The theory of recognition involves the definition of a multi agent ability. A group of agents g can achieve a desired goal if an action a is performed, and group g have the ability to perform a. If an agent can satisfy this condition, then a potential for collaboration exists. An interesting point to note is that CPS states that an agent recognises the potential for collaboration if he *knows of* a group that can cooperate to achieve the desired goal, this group relates to g in the above definition. Knowing of a group that has a multi agent ability excludes the possibility for an agent to attempt to *find* a group of agents who have multi agent ability. The exclusion of the possibility of finding a group with multi agent ability has implications that are discussed in Section 4.

2.5.2 Team Formation

Having identified the potential for collaboration at the recognition stage, the agent knows a group of peers which he believes to have the ability of achieving the desired goal. The purpose of the Team Formation stage of the CPS model is to bring about a mental state within the group in which there is mutual belief (as defined by [2]) that the group can (and will) achieve the desired goal. This belief is known as a joint commitment to the collaborative action.

The joint commitment obtained in the Team Formation stage only ensures that each member of the team is interested in achieving the specified goal. No commitment to specific actions of which will achieve the goal have been made. The Plan Formation stage of the CPS model aims to bring about a state in which each agent in the group knows his responsibility to a specific action and is committed to performing that responsibility.

2.5.3 Plan Formation

There may be many courses of action that could be performed to achieve the goal in question. During the Plan Formation stage the group of agents negotiate to determine which course of action they will perform in order to achieve the goal. Negotiation allows an agent to make reasoned argument s for and against each course of action [19]. Sometimes and agent may not agree with a course of action because of some external consequence it may have, even though it will also achieve the goal in question.

Negotiation of a plan may fail: the group of agents may not be able to reach agreement on a course of action to achieve the collective goal [19]. If negotiation fails, it is likely that the team formation stage will have to be repeated. If a course of action is decided upon, negotiation has been successful and the team can move to the final stage of the CPS model: Team Action.

2.5.4 Team Action

When a team enters the Team Action stage of the CPS model, each agent in the group is committed to achieving the given goal and intends to perform their assigned action as determined in the Plan Formation stage. Each agent also has commitment to notify the rest of the team if they can no longer perform their assigned action or if they receive information which leads them to the realisation that the goal is no longer achievable. This responsibility is derived from the concept of a Joint Intention defined by [2].

3 Scenarios

Selective Data Replication is a relatively new and developing area of research and at present only a small amount of literature on the subject exists. Collaborative updates are one of the more attractive properties of selective data replication, however other than their proposal in [13], no work has been done in defining what it is to update a replication collaboratively. For this reason, a definition of how data consumers collaborate to update a data set replication needs to be derived. In this section, three scenarios of collaborative replication updates are derived in the context of GIS data in a local government setting.

The correctness of these scenarios will not be considered, and future work may be necessary to derive how the greatest benefits can be achieved from collaborative updates. The scenarios will however serve as a guide to the implementation of a framework for achieving collaborative updates. A suitably derived framework will allow the update strategy to be altered without major changes to the framework itself.

3.1 Context

Christchurch, New Zealand is a city of approximately 400,000 residents, the cities local government is known as the Christchurch City Council (CCC). The CCC are responsible for city wide planning, construction, and maintenance of all common types of infrastructure, including roading, water supply, and power supply.

The infrastructure records of the CCC are described and maintained in a large GIS which is broken in to data sets by the infrastructure type (a roads data set maintains information about roading infrastructure). Each type of infrastructure is described by its set of geographical features (roading infrastructure is described by such things as the geographical points which make up the centre line of the road, the set of traffic control devices such as traffic lights and stop signs). When the CCC tender a contract for construction or maintenance of the cities infrastructure, the data set which maintains a record of the affected geographical feature is annotated to indicate that work is being performed (the centre line points of roads are annotated to indicate that road works are under way along that stretch of road).

Geographic information describing the cities infrastructure is valuable not only to the CCC as a way of keeping records, but also to various organisations for planning purposes, and to businesses looking for opportunities to expand. For this reason the CCC make their GIS data sets available to interested parties.

Christchurch is served by the local arm of the New Zealand Emergency Services, this group provides Police, Fire, and Ambulance services to the city. The Christchurch emergency services pride themselves on their rapid response times when called out. Such efficient dispatch times are achieved partly through a system of preplanning the routes that emergency vehicles should take to an accident scene. Each service (Police, Fire, and Ambulance) has a different set of requirements for the route taken to an emergency (the Fire service needs to know the location of fire hydrants) therefore each plans its route individually.

Planning emergency routes involves the consideration of a number of factors to ensure the chosen route is minimally effected by events that could slow the emergency vehicle down. A major factor that is considered in route planning is the location of roadworks around the city, routes must always avoid roadworks as such interference can slow the response time by minutes.

The roading data set provided by the CCC provides the emergency services with the information required to plan emergency routes. Using this data set routes can be determined which are unaffected by roadworks. To allow efficient, timely access to this data, the emergency services store a replicated copy of the roading data set locally. Whenever the CCC make any changes to the roading data set, notifications are sent to each emergency service. To ensure that emergency routes are always up to date with respect to the current state of roadwork interruptions, the replicated data sets stored by the emergency services are updated whenever a change occurs in the roading data set.

Christchurch Pipe Services (CPS) is a local company which owns the water supply pipe maintenance contract with the CCC. Currently water supply pipes are monitored electronically to identify ruptures in pipe lines. When the monitoring system identifies a rupture in a water supply pipe, a notification is sent to CPS informing of the location of and the estimated severity of the rupture. When CPS receive such a notification, a maintenance worker is dispatched to begin repair work.

Most water supply lines are accessible from the cities underground stormwater system, which is large enough for workers to get down to and perform repair work. Access to the stormwater system is provided through manholes spread around the city. Christchurch Pipe Services need to keep track of the locations of these manholes, so maintenance workers can access and begin repair work on supply pipe ruptures as quickly as possible.

The water supply data set provided by the CCC provides Christchurch Pipe Services with information regarding the location of manhole access to the storm water system, as well as detailed information about the pipe work they are contracted to maintain. Because of the frequency at which the company accesses this data set, they store a replicated copy of the water supply data locally. CPS are notified of any changes made to the data set by the CCC, and updates to the replication are performed in line with the update policy employed.

The described environment reflects that of a selective data replication community and provides a platform for creating update collaboration scenarios. As described in Section 2.1, a selective data replication community consists of data providers and data consumers. In the given environment, the CCC acts as a provider, while the three emergency services and the CPS are data consumers of the community. The three emergency services replicate the roading data set, with the fire service also having an interest in the water supply data set to identify hydrant locations along emergency routes. CPS replicates only the water supply data set. Table 3.1 provides a summary of the community using a notation derived from that defined by [13]

P_{CCC}	The Christchurch City Council data provider
$\lambda_{Roading}$	The roading data set provided by the CCC
$\lambda_{Water\ Supply}$	The water supply data set provided by the CCC
C_{Police}	The Police service data consumer
C_{Fire}	The Fire service data consumer
$C_{Ambulance}$	The Ambulance service data consumer
C_{CPS}	The Christchurch Pipe Services data consumer
$\delta_{Roading,\ Police,\ T}$	C_{Police} replication of data set $\lambda_{Roading}$ at time T
$\delta_{Roading,\ Fire,\ T}$	C_{Fire} replication of data set $\lambda_{Roading}$ at time T
$\delta_{Water\ Supply,\ Fire,\ T}$	C_{Fire} replication of data set $\lambda_{Water\ Supply}$ at time T
$\delta_{Roading,\ Ambulance,\ T}$	$C_{Ambulance}$ replication of data set $\lambda_{Roading}$ at time T
$\delta_{Water\ Supply,\ CPS,\ T}$	C_{CPS} replication of data set $\lambda_{Water\ Supply}$ at time T

Table 3.1: The actors and objects of the defined selective data replication community

3.2 Scenario One

In 2006 the CCC city planners identified the need for major extensions to the accessibility of the city from the south. After an extensive analysis of the traffic flows along the current southern routes, an upgrade plan was decided upon that would increase South Road from four to six lanes wide. A contract for the upgrade was put to tender and the successful applicant negotiated to start work on the 1st February 2007. A week before work started on the upgrade the CCC updated the roading data set, adding annotations to the centre line data points of South Road to indicate that roadworks were underway. Notifications were sent to all registered consumers of the roading data set by P_{CCC} to inform of the changes.

C_{Police} received the update notification from P_{CCC} and determined that an update of their local copy $\delta_{Roading,\ Police}$, of $\lambda_{Roading}$ was required. In determining that an update was required, C_{Police} also identified a potential for collaboration in achieving the update. C_{Police} identified C_{Fire} and $C_{Ambulance}$ as potential collaboration partners, as they were also interested in $\lambda_{Roading}$. C_{Police} contacted C_{Fire} and $C_{Ambulance}$ with a request to form a team in order to perform a collaborative update of $\lambda_{Roading}$, both responded with an expression of interest. The team negotiated that C_{Police} would retrieve the updated data set and once complete would notify C_{Fire} and $C_{Ambulance}$. When C_{Fire} and $C_{Ambulance}$ received the update complete notification, both retrieved the updated data set from C_{Police} and the collaborative update was completed.

3.3 Scenario Two

After the annual analysis of troubled intersections around the city, CCC identified the intersection of Oxford and London Streets as being notoriously bad for head on collisions. After consultation with roading engineers a contract was tendered to install traffic lights at the intersection and realign the approach of both Oxford and London Streets. When the successful tender was found for the contract, the CCC updated the roading data set to reflect the roadworks that were about to begin by annotating the centre line data points of the effected stretches of Oxford and London Streets. P_{CCC} sent notifications of change to all consumers of the roading data set as soon as the annotations were complete.

C_{Fire} received the update notification from P_{CCC} and determined that a collaborative update could be performed in order to synchronise their local $\delta_{Roading,\ Fire}$, with $\lambda_{Roading}$. C_{Fire} identified C_{Police} and $C_{Ambulance}$ as potential collaboration partners through their interest in $\lambda_{Roading}$. A request for update collaboration was sent to the potential partners, both responded with an expression of interest. Between the three collaborators it was decided that each would retrieve a subset of the updated $\lambda_{Roading}$, and the subsets would then be distributed between the team. C_{Fire} was assigned the subset defined by street names starting with 'A' through to those starting with 'I', C_{Police} was assigned the subset defined by street names beginning with 'J' through to those starting with 'P', $C_{Ambulance}$ were responsible for the remainder of $\lambda_{Roading}$ being street names beginning with 'Q' through 'Z'.

3.4 Scenario Three

At the end of 2006, a large subdivision was completed in the north of Christchurch. As sections in the subdivision started to sell and residents started to move in, the CCC began to notice a larger than expected demand on water supply lines. After a review of the possible options to satisfy the increased demand, it was decided that a major upgrade of the main water supply line to the north of Christchurch was required. The CCC put a contract to tender for the upgrade, and the successful applicant agreed to begin work on the 1st of March 2007. Once plans were completed, and before work on the upgrade began, the CCC updated the water supply data set to reflect the upgrades, and sent notifications of change to all consumers of the data set.

C_{CPS} received the update notification from P_{CCC} regarding the changes to $\lambda_{Water\ Supply}$ and determined that an update of their local copy $\delta_{Water\ Supply,\ CPS}$ was required. C_{CPS} identified a potential for collaboration with C_{Fire} to achieve the update, and sent a collaboration request. C_{Fire} responded with an expression of interest. C_{CPS} and C_{Fire} determined that a local proxy should be set up to store a shared replication of $\lambda_{Water\ Supply}$, with C_{CPS} responsible for updating the subset of $\lambda_{Water\ Supply}$ which described pipe lines and manhole points, and C_{Fire} responsible for the subset describing hydrant locations.

4 Formalization

The Cooperative Problem Solving theory (CPS) defines a four stage model which can be used to achieve collaborative action within a multi-agent system. The stages of the CPS model have a sequence, and therefore there is a beginning and an end to the collaborative process, something that is less evident in the high-level theories described in Section 2.4. Because of the process that is defined by the CPS theory, it seems like it provides a suitable model for implementation of collaborative action: there is a start and a finish. Each stage of CPS has been derived from work on formal collaboration theories such as that examined in Secion 2.4. The model CPS provides is therefore still a high-level abstract specification of collaboration, which requires interpretation for an implementation using a specific platform for a specific domain.

The aim of this project is to develop a collaboration framework using the OPAL agent platform, that allows data consumers to perform collaborative updates in a selective data replication community. This section provides a formalization of the CPS model, by overcoming its inherent abstraction and describing the steps that should be implemented for each stage of the CPS model to allow collaborative updates to occur. UML Activity Diagrams [14] have been derived using ideas from both CPS and the collaboration theories examined in Section 2.4, to define the flow of tasks at each stage of the formalization of CPS. The definition of these tasks has been guided by the architecture of the OPAL agent platform, and the Scenarios for collaborative updates developed in Section 3.

4.1 Recognition

As described in Section 2.5.1, the Recognition stage of CPS begins with an agent individually coming to believe there is potential for collaborative action in order to achieve a goal that it holds. In the case of selective data replication, the goal will always going be to update a data set the consumer agent replicates to reflect changes that have occurred at the source of that set. Potential for collaboration exists if the consumer knows of a group of his peers that have the ability to perform a collaborative action that will achieve that update.

The need for a replication update is triggered by the notifications of change a data producer provides to all registered data consumers when a change occurs in the data set that producer is responsible for. When a consumer receives such a notification reasoning must be performed to determine the effect of the change on its personal goals. Such reasoning may lead to the conclusion that an update is required and that the update could be achieved collaboratively. The result of reasoning is determined by the update policy the agent employs (described in Section 2.1), if it is determined that a collaborative update may be appropriate the next action the agent must take is to determine a group of his peers who have the ability to collaborate.

By definition the CPS theory states that for there to be potential for collaboration, the agent who holds the goal that is believed to be achievable through collaborative action, *knows of* a group who can jointly achieve that goal. The authors recognise however, that this is ”an overstrong assumption“ [19] and cite that allowing an agent to *attempt to find* the identity of a group that can achieve the goal would complicate the definition of their theory. Catering for this possibility is noted as an area for future work. Enforcing such an assumption in the case of selective data replication will increase the required ”memory“ load on each consumer and is likely to reduce the scope of benefits gained from collaborative updates. Forcing a consumer to remember all possible combinations of his peers who it believes are capable of performing a collaborative update, greatly increases the beliefs it holds regarding the community. As more and more consumers join the community through subscribing to some data set provided within, the rate of increase of required beliefs will is likely to be combinatorial. At the same time, by restricting an agent to the groups it has a belief in, the possibility is removed for collaboration with larger aggregate groups or with consumers who have recently joined the community and are yet to make contact with the initiating agent.

Due to the advantages gained from allowing a consumer to attempt to find a group capable of performing a collaborative update, this is the technique that has been implemented in the proposed collaboration framework. It will be considered that if a consumer can find a group that is capable of collaborative action, potential for collaboration has been recognised. The following sections describe the factors that are considered and the approach taken in identifying such a group.

4.1.1 Common Interest

Once a data consumer has made the decision to attempt a collaborative update, it must search for other consumers who may be willing to participate in the collaborative action that will achieve that update. The fundamental consideration of the search is the particular interests of potential collaborators: there is not much chance that a consumer who does not replicate the data set in question will be willing to collaborate. The goal of the search is therefore to identify a group of consumers with a common interest in the data set that will be targeted by the collaborative update. Formally, common interest between two data consumers can be defined as:

> Data consumers C_A and C_B have a common interest in a data set λ_C provided by producer P_C, if both C_A and C_B have a subscription with P_C to receive notifications of change when the contents of λ_C are altered.

4.1.2 Collaborative Ability

When a group of consumers has been found who have a common interest in the data set requiring an update, the initiating agent must then determine the collaborative abilities of the members of that group. Any consumer with such ability will become a potential member of the team who will perform the update. At this point, no update strategy has been chosen and the aim is to simply identify which agents have the ability to join a collaborative team.

The initiating agent must make contact with the consumers that have been identified as having a common interest. A request is made for the collaborative ability of each with respect to a collaborative update of the data set in question. The response contains an indication of the ability of the agent to collaborate, and the update strategies that it is capable of participating in. CPS uses the Plan Formation stage to determine the collaborative strategy and tasks that each member of a team will perform. By having potential team members include the strategies they are capable of participating in as part of a response to their ability, the negotiation required during Plan Formation is simplified (as is discussed in Section 4.3).

4.1.3 Recognition Process in UML

Figure 4.1 shows the entire process of the Recognition stage of CPS as defined for collaborative updates in a selective data replication community. The process is initiated when an agent receives a notification of change regarding a data set it is subscribed to. This change is then assumed as a belief of the agent, and reasoned about against the update policy it employs. If it is determined that an update is required and the possibility for collaboration exists, a template is created that can be used to search the Agent Directory Service for a group of agents with a common interest. The initiating agent then requests the collaborative ability of each agent it has found to have a common interest, any consumer who responds with a collaborative ability is added to a group of possible team members. The initiating agent must then evaluate the abilities of that group and determine if together, they are capable of executing a collaborative update strategy.

Having found a group of agents who together have the ability to carry out a collaborative update, the initiating agent has satisfied the requirements of the CPS definition of Recognition: a group has been found who the initiating agent believes to have the ability to perform a collaborative update which will achieve the goal of updating a data set replication to reflect the changes that have occurred at its source. The process will now move to the Team Formation stage with the initiating agent taking on the role of team leader.

This formalization of the Recognition stage for collaborative updates has extended the CPS theory definition by allowing an agent to attempt to find a group with the ability of achieving the update together. The authors of CPS do not rule out this strategy, however it is left to future work for its complication of the definition. The technique that has been used to achieve this extension has been guided by the way in which an agent solicits assistance in the Planned Team Activity collaboration theory [9], described in Section 2.4.3.

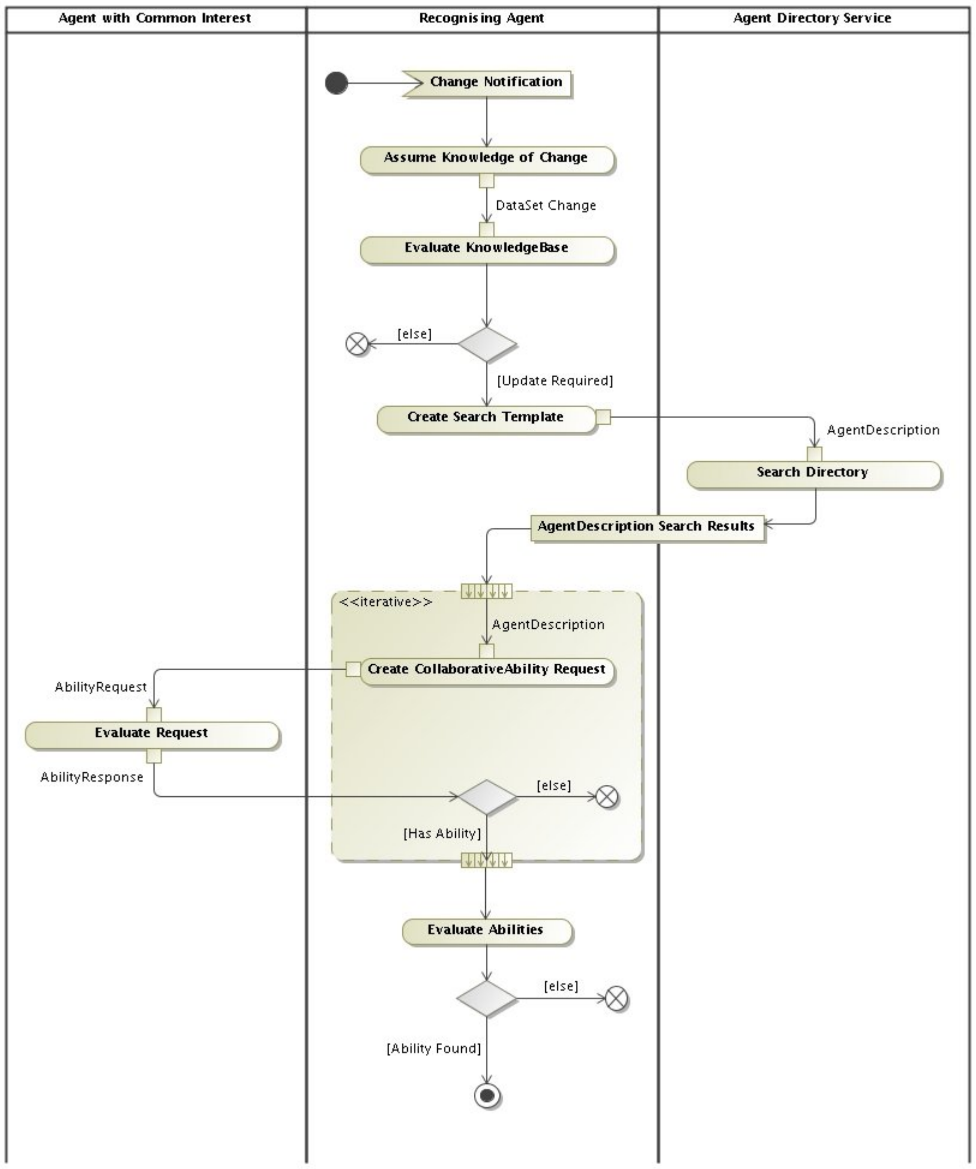

Figure 4.1: UML Activity diagram representing a formalization of the process of Recognition

4.2 Team Formation

Section 2.5.2 identifies the conclusive criteria of the Team Formation stage of CPS as bringing about a joint mental state within a group of agents capable of performing the required collaborative action, in which each agent believes that the group can achieve the desired goal, and each agent is committed to team action with respect to that goal. Once a data consumer has recognised the potential for collaboration to achieve the goal of updating a replicated data set, it must solicit assistance from the group it believes to have the required collaborative ability. If this attempt is successful a team of consumers will exist who are committed to performing a collaborative update of the changed data set. A team leader may not be successful in creating a team: although the group of consumers that were found during Recognition have the ability to collaborate, they may not have an interest in the team leaders goal of a replication update. This results in two main tasks that the team leader must perform during the Team Formation stage: it must first determine which of the consumers it identified as having the ability to collaborate also have an interest in collaborating; it must then bring the consumers which have an interest in collaborating to the belief that they are part of a team that is committed to performing the collaborative update.

4.2.1 Collaborative Interest

Before the team leader can create a team of consumers, it must identify which of the agents it knows to have the ability to collaborate also have an interest in the goal of achieving a replication update. Such an interest will be determined by each potential team member reasoning about the current state of his replication against the update policy that is employed. In order for the team leader to determine whether this interest exists, a request of interest is sent to each potential team member. When a potential team member receives the request it will determine its position in regards to the data set which is to be updated, and then inform the team leader of any interest in joining the team. The team leader will add all consumers who have an interest in being part of the collaborative update to a group who will be requested to join the team, consumers lacking an interest are discarded.

A lack of interest could result from the following situations: the agent has recently updated his replication either individually or as part of some other group, and does not require another update as the amount of change that has occurred since that update is not considered significant; the agent believes that performing the update with the group requesting his assistance may have a detrimental effect on another of his goals e.g. the resources needed to collaborate with the group may be required for some other task the agent is currently or is about to undertake. The first of these cases is determined by the update policy the consumer employs, and is therefore specific to SDR. The second case is determined by a resource management policy and will be discussed as future work.

4.2.2 Creating a Team

In order to create the joint mental state required by the CPS theory, the team leader attempts to make each consumer who has an interest in the collaborative update believe that it is a member of an update team. The belief of being in an update team is represented by the consumer taking on the role of team member. The team member role allows the consumer to remember the identifier for the team (assigned by the team leader), and the other consumers who are part of that team. The consumer will not yet have taken on any responsibility towards individual tasks of the collaborative action, such details are determined during the next stage of CPS, Plan Formation.

The team leader will send a join team request to each consumer who responded with an interest in joining a team to perform an update collaboratively. The join team request includes the team identifier and locators for each of the consumers who have been added to the team. On receiving the request, the consumer assumes the role of team member and commits to performing an update as part of that team. Such a commitment is represented by the agents belief that it is a team member, consequences of this belief will be identified in Section 5.3.

4.2.3 Team Formation Process in UML

Figure 4.2 shows the actions that must be performed by both the team leader and his potential team members which were identified in the Recognition stage, in order to create a team of consumers who are committed to performing a collaborative update. Each of the potential team members is sent an interest request by the team leader, the member must evaluate that request and determine if it is interested in the collaborative update being proposed. The potential team member then replies to the team leader informing of his position in regards to the request, if it is interested the team leader adds him to the team, if not it is discarded. The team leader must then send join team requests to each consumer who was added to the team. This request is formal recognition that the consumer has been added to the team and should take up the role of team member. Once the team leader has had a reply from all his team members informing that they have taken on the belief that they are now part of the team, the Team Formation stage is complete.

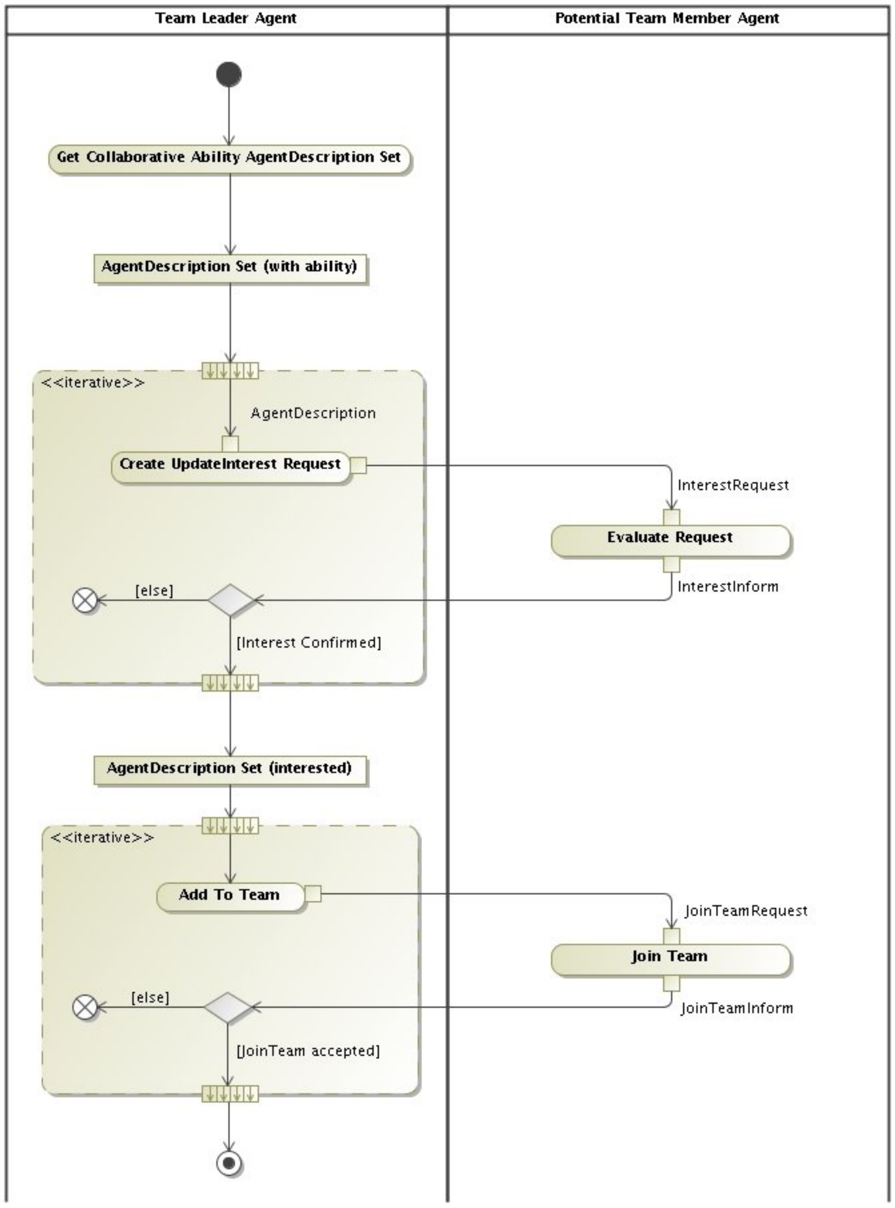

Figure 4.2: UML Activity diagram representing a formalization of the process of Team Formation

4.3 Plan Formation

Successful execution of the Team Formation stage results in a team of consumers who have a commitment to collaborative action, however the individual tasks of that action must be assigned so that each member knows their personal responsibility. The Scenarios derived in Section 3 define three alternative strategies that data consumers can use to update collaboratively. Each strategy has a series of tasks that must be executed by one or more consumers in order to achieve the update. The Plan Formation stage defines how these tasks are assigned to individual members of the team.

The core activity of the Plan Formation stage is negotiation. Members of the team must negotiate a plan which both achieves the teams shared goal and satisfies the individual requirements of each member. The scenarios of Section 3 outline three different update strategies that will all achieve the goal of retrieving a data set update, these strategies are the plans of SDR. In Section 4.1 it was stated that in replying to a request of ability, a consumer would provide the update strategies it is capable of participating in so that negotiation during Plan Formation is simplified. Because the team leader knows which plans his team are capable of carrying out, the required negotiation is that which determines the individual tasks each team member will perform, the plan that will be used can is determined by the team leader based on the capabilities of team members.

Negotiation is a complex task, for this reason the authors of CPS avoid a formalization of what it is to negotiate a plan 2.5 and instead refer the reader to preliminary work on models of argumentation [10]. The task of implementing negotiation provides further problems due to the requirement of some form of artificial intelligence. The solution that has been used for the proposed collaborative framework, is to provide contract-tender capabilities to consumer agents. A similar approach was used successfully by [16] in an agent based e-commerce setting. The contract-tender solution is described in the following sections.

4.3.1 Collaborative Contracts

Contracts are used by the team leader to notify his team of the tasks that make up the collaborative plan. The team leader determines the update strategy that will be used through an evaluation of the capabilities of his team. He has knowledge of each team members capability from the responses he received to his request for collaborative abilities during the Recognition stage. In the case of a team being capable of multiple update strategies, the team leader simply tries the first strategy in the list. This is a simplifying assumption however, and a discussion of the implications and changes that may be required can be found in Sections 6 and 7.

Once an update strategy has been determined, the team leader must create a contract which includes the tasks required to be executed in order to achieve the collaborative action. Each team member is sent a contract and expected to evaluate the tasks it includes and provide the team leader with a tender. For each update strategy that a consumer is capable of participating in, it must have knowledge of the tasks that are required of that strategy and the individual actions it must take to perform those tasks.

4.3.2 Tendering for Tasks

Tenders are used by team members to negotiate for tasks that must be performed as part of the collaborative action. For each task that is defined in the contract that the team leader provides, a team member must identify the task and then determine the cost it would incur in executing the task. A tender is then created containing the tasks that the agent wishes to apply for and the cost at which that task can be executed, the tender is sent back to the team leader to allow for task delegation.

Execution costs for individual tasks could be effected by a number of factors such as the resources required, the size of the team, and the importance of the task. Determining precisely which factors have an effect and the weight each provides on the overall cost of performing the task is beyond the scope of this project. A discussion of the insights found in relation to determining these costs appears in Section 7

4.3.3 Assigning Tasks

Once the team leader has received tenders from all members of the team (and has created its own tender so it is considered in the assigning of tasks), tasks must be assigned to team members. Each member has provided a tender which includes the tasks it wishes to perform and a cost for performing that task. The team leader evaluates the received tenders, and assigns each task based on the execution costs that have been proposed by the team members.

If a particular task required by the chosen update strategy is not tendered for by any team member, one of two situations will occur: if the team leader identified multiple strategies that the team could perform, it retries using the next strategy on the list; if no other strategy is available to the team, the Plan Formation stage has failed, and the team leader must inform the team that a plan for collaborative action could not be created.

4.3.4 Plan Formation in UML

Figure 4.3 shows the tasks performed by the team leader and the members of the team during the Plan Formation stage. The team leader creates an update contract request and sends a copy to each member of the team. When the request is received, the team member must evaluate the tasks included in the contract and prepare a tender containing the tasks it wishes to be assigned to and the cost of execution. When the team leader has received all tenders, an evaluation determines which member should be assigned to which task. If a required task is not tendered for, the team leader creates a new update contract for an alternative update strategy and starts the process again. Tasks are assigned to team members through an assume role request, on receiving such a request the member must take on the belief that he has be assigned to the specified task. That belief provides a commitment to execution of the assigned task at the required time.

4.4 Team Action

If a team is successful in negotiating a plan and assigning individual tasks of that plan to team members, it is expected that the team will then execute the collaborative action for which they were formed. The Team Action stage is simplistic and only requires team members to execute the tasks that were assigned to them during Plan Formation.

4.4.1 Team Action in UML

Figure 4.4 is a UML representation of the Team Action stage of the proposed framework. The team leader simply initiates action by sending an execute task request to each member of the team. This initiation is used to ensure that team members do not start task execution until all tasks have been assigned in the previous stage.

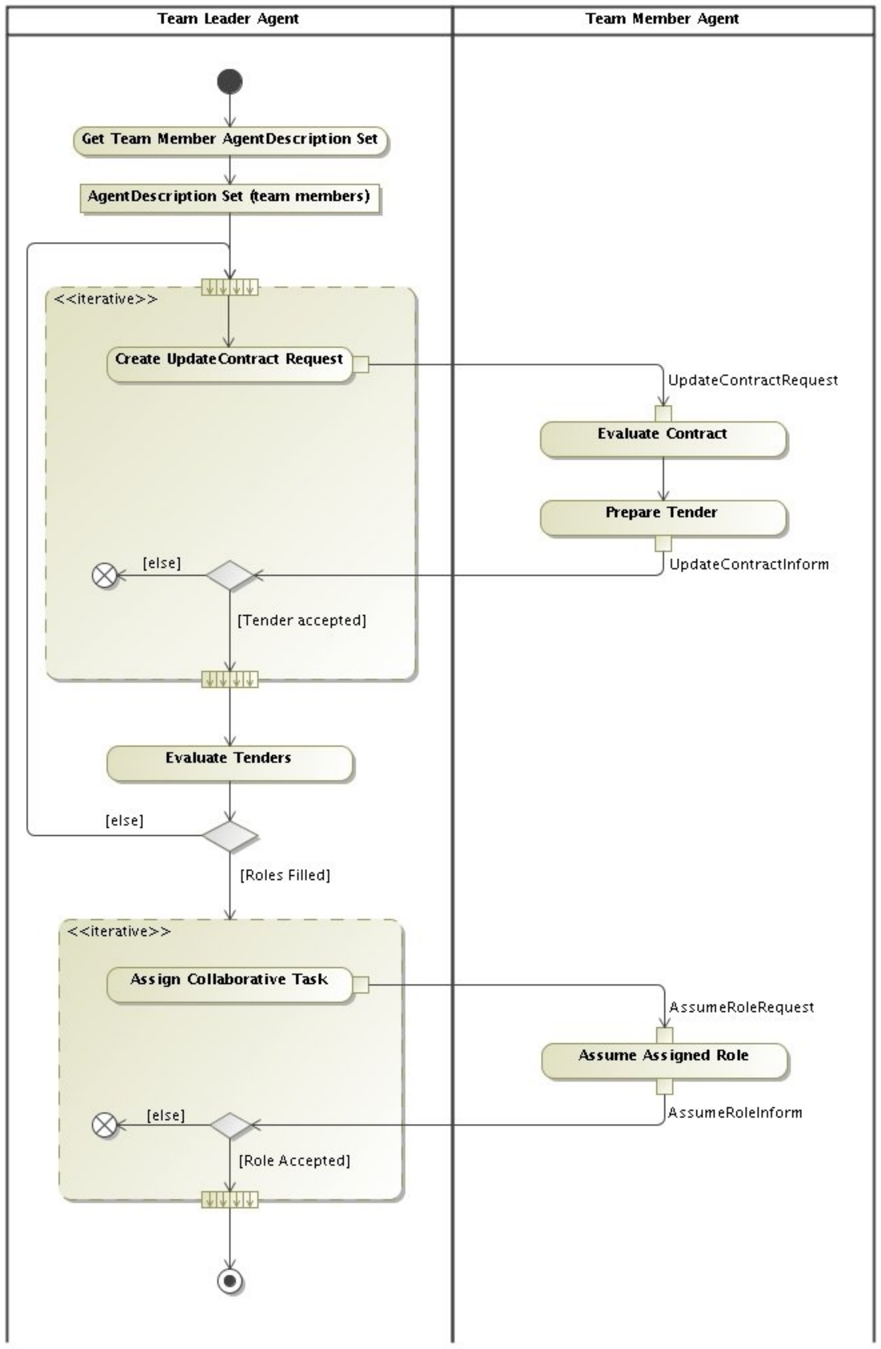

Figure 4.3: UML Activity diagram representing a formalization of the process of Plan Formation

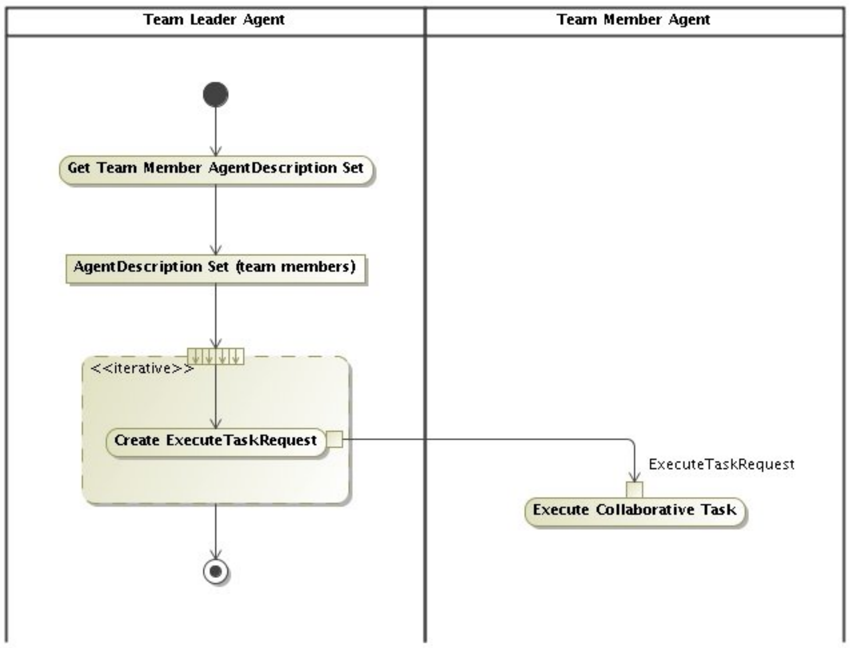
Team Leader Agent
Team Member Agent
Get Team Member AgentDescription Set
AgentDescription Set (team members)
<<iterative>>
Create ExecuteTaskRequest
ExecuteTaskRequest
Execute Collaborative Task

Figure 4.4: UML Activity diagram representing a formalization of the process of Team Action

5 Implementation

Section 4 presented a formalization from the abstract description of the CPS model to a series of tasks that can be executed by data consumer agents. UML activity diagrams were presented to describe the flow between tasks, and the interactions between the collaborating agents. Implementation of the framework has required that a suitable solution be created for each task identified by the activity diagrams. This section provides a discussion of some of the important aspects of the implementation of these tasks, and the way in which features of the OPAL platform have supported the derived framework.

5.1 The Search for Common Interest

In Section 4.1.1 the concept of common interest was introduced and identified as being the fundamental consideration in searching for potential collaboration partners. Common interest can be identified through the subscription a consumer has to a data set of interest. By searching for consumers who are subscribed to the data set that must be updated, a group of potential collaborators with a common interest can be found.

When a consumer subscribes to a data set he assumes the SubscriptionRole through creation of a micro-agent which implements that role. This role provides the agent with several capabilities required of a data consumer subscribed to some data set. Of interest to the task of searching for common interest, is the description an agent playing the SubscriptionRole registers with the OPAL platform. When a SubscriptionRole is created for an agent, a description of the data set he subscribes to is created in the form of a DataSetSubscription. The DataSetSubscription stores details of the subscribed to data set, and is included in the description that is registered with OPALs Agent Directory Service (ADS, introduced in Section 2.3) with the key “dataSetInterest”.

The OPAL platform provides registered agents with the capability to search the ADS in order to find agents with particular characteristics. A consumer who is interested in finding peers with a common interest in a particular data set creates a template DataSetSubscription which represents that data set. The consumer then adds the DataSetSubscription template to a search template which is provided by OPAL for the task of searching the ADS. A search request is made, and the ADS will return all of the AgentDescriptions that have been registered with a matching DataSetSubscription to that included in the search template.

The OPAL implementation of the ADS search first matches for type i.e. if the search template contains a DataSetSubscription then all registered AgentDescriptions that have a DataSetSubscription will be chosen. The Java equals function is then used to determine any matches. This implementation is somewhat restrictive, the contents of a search template must be of a type that is shared between all consumers, and the equals function of that type must be appropriately overridden. Due to time constraints, further work in the area of subscription representation was not possible, however in Section 7 a more desireable solution is discussed.

5.2 Requests and Informs: Consumer Communication

From the activity diagrams derived in Section 4 it can be seen that a large amount of communication is required to achieve collaborative action. Communication consists of the team leader making requests of team members, and those members then replying via an inform. OPALs messaging implementation restricts the content of messages to key-value string tuples. While this restriction supports platform and implementation independence, it makes the task of transferring complex structures such as collaboration contracts difficult. A significant level of message processing and parsing is required at both sender and receiver ends of a communication channel.

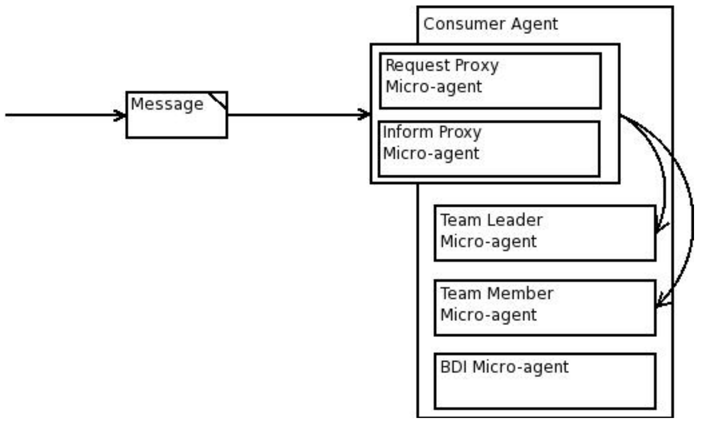

Figure 5.1: Consumer agent structure showing the interface provided by the proxy micro-agents

In order to isolate the required parsing and processing, each consumer agent uses proxy micro-agents which intercept incoming messages, perform the required processing, and forward the result to the intended receiver (in most cases either the team leader or team member micro-agent). Figure 5.1 shows the way in which these proxy micro-agents provide an interface to other consumers in the community. This solution allows each consumer to use an independent implementation of constructs such as collaboration contracts, as long as the protocol used for messaging is understood and can be parsed into the required format.

A high level communication protocol has been derived to allow members of the consumer community to share a common understanding of the messages that can be received. The protocol is required to allow consumers to determine the meaning and usage of each key-value pair used in the different message types. When a message is parsed from the OPAL form in which it arrives, the protocol is used to determine which keys will provide the required information. The following two sections provide details of the communication protocol.

5.2.1 Request Protocol

Tables 5.1 - 5.4 describe the keys used in each request message type, and a description of the content for each. Ability and Interest requests use the same set of keys, therefore a table for Interest requests is omitted.

Key	Description of Value
SETNAME	the name of the data set this ability request regards
SETPRODUCER	the producer of the data set this ability request regards
REQUESTSOURCE	the agent who requested the ability (team leader)

Table 5.1: Ability request protocol

5.2.2 Inform Protocol

Interest informs use the same set of keys as Interest requests with an addition “INTERESTED” key which is an indication of the consumers interest in collaborating (i.e. will be either true or false). Join Team informs are simply an acknowledgement that the consumer has joined the team and so only include a “TEAMIDENTIFIER” key. Assume Role informs use the same set of keys as Assume Role requests. Tables 5.5 and 5.6 show the protocols used for Ability informs and Evaluate Contract informs (the tender

Key	Description of Value
TEAMLEADER	the consumer leading the team that should be joined
TEAMIDENTIFIER	the team identifier of the team that should be joined
TEAMMEMBERCOUNT	the number of team members
TEAMMEMEBER0 → TEAMMEMBERi	the identity of each team member where i is the value of TEAMMEMBERCOUNT

Table 5.2: Join Team request protocol

Key	Description of Value
TEAMIDENTIFIER	the team identifier of the team the contract relates to
ROLECOUNT	the number of tasks the contract defines
ROLE0 → ROLEi	the identifier of each task where i is the value of ROLECOUNT

Table 5.3: Evaluate Contract request protocol

the agent replies with).

The execute task message that is sent to team members to initiate team action (as can be seen in Figure 4.4) has been omitted here as it does not carry any information and a protocol was not required. It is simply a directive to inform members that action can begin.

5.3 Leading and Joining a Team using Roles

Two important concepts introduced in the formalization of Section 4 were that of a team leader and a team member. The team leader has the role of directing the creation of the team, and overseeing the negotiation of task assignment. Team members must respond to requests of the team leader, and both negotiate for and execute tasks that make up the collaborative action. Taking on either of these roles results in two consequences: the agent must provide the behaviours that are associated with playing that role; in playing such a role, the agent becomes aware that he is part of a team and takes on additional responsibilities toward the team members.

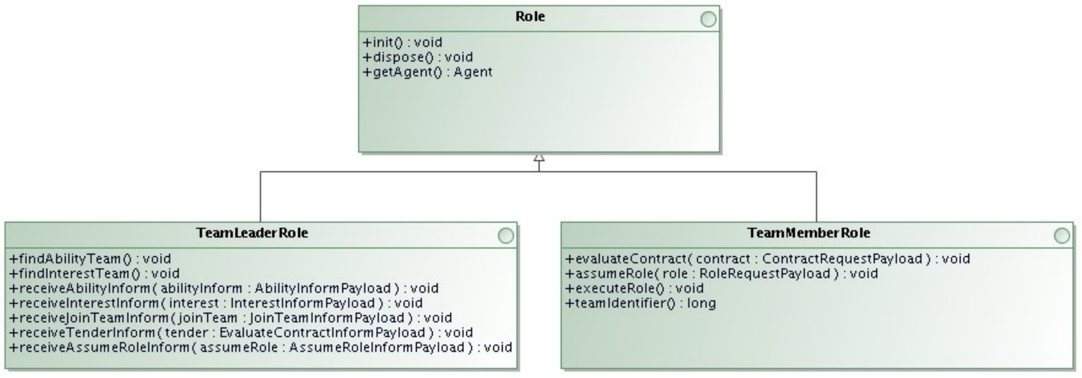
Role
+init() : void
+dispose() : void
+getAgent() : Agent
TeamLeaderRole
+findAbilityTeam() : void
+findInterestTeam() : void
+receiveAbilityInform(abilityInform : AbilityInformPayload) : void
+receiveInterestInform(interest : InterestInformPayload) : void
+receiveJoinTeamInform(joinTeam : JoinTeamInformPayload) : void
+receiveTenderInform(tender : EvaluateContractInformPayload) : void
+receiveAssumeRoleInform(assumeRole : AssumeRoleInformPayload) : void
TeamMemberRole
+evaluateContract(contract : ContractRequestPayload) : void
+assumeRole(role : RoleRequestPayload) : void
+executeRole() : void
+teamIdentifier() : long

Figure 5.2: UML Class diagram of the Role interfaces for a Team Leader and a Team Member

The micro-agent architecture and role capabilities provided by OPAL have been exploited in implementing the team leader and team member concepts. From the activity diagrams developed during the formalization process, the major tasks of both leaders and members were identified and OPAL roles were created which provide the interface to those tasks. Figure 5.2 shows a class diagram of the resulting role definitions. The

Key	Description of Value
TEAMIDENTIFIER	the team identifier of the team the role should be played for
ASSIGNEDROLECOUNT	the number of roles that have been assigned to the receiving consumer
ASSIGNEDROLE0 → ASSIGNEDROLEi	the identifier of the role that has been assigned

Table 5.4: Assume Role request protocol

Key	Description of Value
SETNAME	the name of the data set this ability inform regards
SETPRODUCER	the producer of the data set this ability inform regards
STRATEGYCOUNT	the number of strategies in the message
STRATEGY0 → STRATEGYi	the identity of each update strategy where i is the value of STRATEGYCOUNT

Table 5.5: Ability inform protocol

implementation of these roles is provided by micro-agent classes for a team leader and for team members.

The OPAL implementation of micro-agents allows dynamic loading of micro-agents that play a specific role at runtime. This allows consumers to take on a role as either team leader or team member as a result of a request or after identifying it as a requirement as a result of some evaluation or reasoning. Team leaders are created at the recognition stage of the collaborative process, as a result of reasoning about a change notification and concluding that a collaborative update may be used to rectify the effect of the change on a replication. Team members are created after a consumer has indicated an interest in collaborating to a team leader, and is requested by that leader to join a team.

As a result of having a micro-agent playing either a team leader or team member role, a consumer is aware that it belongs to a team. The Team Action stage of the CPS model requires that any agent who realises an inability to either perform or complete an assigned task, make his team aware of that inability. The reasons that may cause an agent to come to this realisation are an area of future work (see Section 7), however with this team membership belief in place, it will not be difficult to extend the required activities at the point of such a realisation to include the notification of the inability to the team.

5.4 Individual Tasks of Collaborative Action

The scenarios described in Section 3 outline three collaborative update strategies, and break those strategies down into the individual tasks that are performed by members of the team. These update strategies must be known by any consumer who wishes to collaborate in team action, and the consumer must be able to reason about and evaluate his ability to perform each of the tasks involved. For the purposes of this project it is assumed that each consumer understands all three strategies and has the ability to perform any of the tasks that are involved. The result of this assumption is that all consumers in the community will attempt to be

Key	Description of Value
TEAMIDENTIFIER	the team identifier of the team the tender relates to
ROLECOUNT	the number of team members
ROLE0 → ROLEi	the identity of the role tendered for where i is the value of ROLECOUNT
ROLECOST0 → ROLECOSTi	the cost of executing the role tendered for

Table 5.6: Evaluate Contract inform protocol

included in any collaborative update that occurs within, Section 6 provides a discussion of the implications that may result.

The representation of an agents knowledge of update strategies and the tasks that are required has been implemented through beliefs the consumer holds and rules that are evaluated by the BDI micro-agent. As an example, when a team member receives an update contract containing a task that must be performed, beliefs are checked for the existence of such a task, if that task exists an attempt is made to load an instance of a class that represents that task (described in Section 5.5), if the attempt is successful the instance is inserted in to the consumers knowledge base for cost evaluation. Listing 5.1 shows an example of a rule that is used to determine the cost of executing an update subset task (as in scenario two, Section 3.3).

Listing 5.1: A rule used to determine the cost of executing an update subset task. The format is that which is expected by the ROK reasoning engine

```
rule: ContractScenrioOneRule
{
  declarations:
    TeamMemberMicroAgent agent;
    CostEvaluatorMicroAgent evaluator;
    UpdateSubsetRole update;

  conditions:
    evaluator.cost(update) <= 10;

  actions:
    agent.addRoleToTender(update);
}
```

The implementation of individual tasks of an update strategy uses OPAL roles and micro-agents. The CollaborativeActionRole represents a single task of an update strategy, and defines a single function, executeRole(). Different tasks are created by providing a suitable micro-agent implementation of the CollaborativeActionRole. Figure 5.3 is a UML class diagram showing the two tasks of scenario one, both implement the CollaborativeActionRole and extend the OpalSubAgent class (indicating that they are micro-agents).

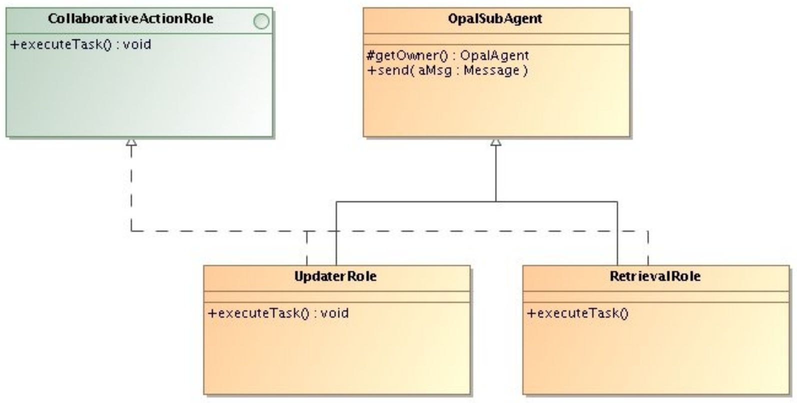

Figure 5.3: UML Class diagram of the Task/Role structure for the tasks of Scenario One

5.5 Collaborative Contract and Tender Representation

Contracts and Tenders are sent between team leaders and members as a way of negotiating the assignment of the tasks that make up a collaborative update strategy. Consumers use a micro-agent which manages the BDI capabilities of the consumer to evaluate the tasks contained in a contract to determine an execution cost for each task. In order to evaluate the tasks in a contract, the tasks are inserted in to the knowledge base of the BDI micro-agent, reasoning is performed against the rule set the agent employs, and the result is a cost for executing the task.

As described in Section 2.2, the ROK reasoning engine uses object instances contained in the knowledge base to trigger the evaluation of its rules. The contract describes tasks using string identifiers, as this is the only representation that an OPAL message will allow. Before a task can be evaluated, the string representation must be converted to an object instance that can be inserted in to the knowledge base of the consumer. Due to time constraints a simplified implementation has been used to achieve this conversion, section 7 describes future work in this area that would lead to an ontology based look up system for loading instances of a task based on an identifier. The implementation created for the purposes of this project uses fully qualified class names for task identifiers, reflection is then used to load an instance of the class with that name. Listing 5.2 shows the code used to achieve this.

Listing 5.2: Code used to load a task instance from an identifier.

```
public void evaluateContract(EvaluateContractRequestPayload contract) {
  Class cls = null;
  CollaborativeActionRole role = null;

  for(String roleName : contract.getRoles()) {
    try {
          cls = Class.forName(roleName);
          role = (CollaborativeActionRole) cls.newInstance();
        }
        catch (ClassNotFoundException e1) {
          System.err.println("Could␣not␣instantiate␣role␣" + roleName);
        }
        catch (InstantiationException e2) {
          System.err.println("Could␣not␣instantiate␣role␣" + roleName);
        }
        catch ( IllegalAccessException e3) {
          System.err.println("Could␣not␣instantiate␣role␣" + roleName);
        }
        evaluator.insertKnowledgeFragment(role);
  }
}
```

6 Results

The Cooperative Problem Solving theory described in Section 2.5 defines a set of steps which the authors claim to exist in "most instances" of collaborative activity, and if followed will take that activity from beginning to end. Using this theory, a framework that is suited to the domain of collaborative updates in selective data replication has been derived and implemented using the OPAL agent platform. This section explores the results of the formalization and the resulting implementation with respect to the scenarios of collaborative updates derived in Section 3.

The discussion of results will focus on two areas of interest: the ability to move from the abstract theory described by CPS to a framework which implements the four defined stages; and the implementation of the individual tasks that make up a collaborative update strategy and the guidance provided by CPS for this process. First however a discussion of the behaviour of the framework when deployed within a community of OPAL agents is provided.

6.1 Framework Deployment

In order to determine if the developed framework allows data consumers to successfully negotiate and carry out collaborative updates, it was deployed within a group of agents which existed within an executing OPAL platform instance. Because the aim of the project was the development of the collaborative framework, details of data producers and the way in which notifications of change are provided to subscribed consumers were ignored, and a mock notification of change generated and sent to a consumer with the belief that a subscription to the data set that notification applied to existed.

All agent communication and the result of actions that consumers take on receipt of specific requests or informs was logged. Appendix A shows a trace of the groups actions when Scenario One is used as an update strategy. The traces for the execution of the other two strategies only differs in the names of roles that are negotiated for, and the output of the actual execution of those roles. The trace shows that the framework successfully takes a group of consumers from the point where an individual realises the potential for collaboration through to a point where all tasks of a collaborative update strategy have been assigned and execution of those tasks occurs. This is the aim of the process that is defined by the CPS theory.

What is not evident from the trace is the importance of the relationship between the individual tasks that make up the collaborative action. In the execution shown in the trace, the ordering of the tasks results in a successful collaborative update being performed. The UpdateRole is executed first by the ambulance consumer, which retrieves the update from the producer agent. The police and fire consumers then request a copy of that update from the ambulance consumer.

In defining collaboration contracts and the tasks which they consist of, no consideration has given to the inter task effects. If in the execution of scenario one, a consumer playing the RetrievalRole attempts to request the update from the consumer playing the UpdateRole before the UpdateRole task has been executed, a problem will occur as there will not yet be an update to request. This situation can occur in all three of the scenarios that were defined in Section 3.

The definition of CPS does not consider the effects that occur between individual tasks of a collaborative action. This assumption is fair, as not all types of collaborative action will require such consideration. The tasks that a collaborative action consist of are entirely domain dependent, and the abstract model CPS defines is independent of such details. This provides an insight in to the way in which collaborative action in SDR must be solved, the three scenarios defined in Section 3 must be explored in further detail, and a suitable representation for tasks within the implemented framework must be found.

6.2 CPS: From Theory to Practice

CPS theory was chosen as a suitable model for creating a framework for collaborative updates due to its definition of collaborative action as a process which has a beginning and an end. The theory is still domain and platform independent and therefore provides a level of abstraction that must be overcome by an implementation. Section 4 attempted to formalize the theory to a process suited to collaborative updates in selective data replication implemented using the OPAL agent platform.

The result of the formalization is a set of activities, defined using UML activity diagrams that are required to move from the beginning to the end of the CPS theory. These activities were defined within the restrictions imposed by OPAL in order to allow a suitable implementation. The subsequent implementation is proof that the abstraction of the CPS theory can be overcome using the OPAL platform to create a framework suitable for use to achieve collaborative updates.

6.3 Update Strategies

Section 6.1 identified the problems caused by the informal nature of collaborative update strategies defined by the scenarios of Section 3. The goal of this project was to develop a framework for achieving collaborative updates in SDR, the scenarios used had to be defined in order to have some goal for what it is to perform a collaborative update. However the importance of a formal definition of what it is to update collaboratively, and how such collaborative strategies should be defined and represented within the framework were overlooked. This has provided a useful insight into where attention needs to be focused in order to suitably achieve collaborative updates and is discussed as future work.
In summary, the framework that has been developed has successfully provided an implementation of each of the stage of the CPS theory. It is possible to move a group of agents to a position where they are ready to perform a collaborative update and executing that update will in some cases be successful. The implementation has highlighted the need for future work to define how collaborating agents should consider the responsibility and relationship an individual task should be represented and implemented. With such a mechanism in place, collaborative updates will be guaranteed successful under normal conditions.

7 Future Work

The formalization of the CPS theory and subsequent implementation of a framework for collaborative updates in selective data replication has exposed the complexity of collaboration in a multi-agent system and has provided major insights for future work. The framework implementation has been simplified in certain areas to allow for timely completion of this work, however it has confirmed that collaborative updates can be achieved using the abstract theories defined by popular collaboration models. This section provides a discussion of the areas in which these simplifications have been made, and the direction in which future work may be beneficial.

7.1 Agent Directory Search

Section 5.1 described the implementation of the technique a team leader uses to identify a group of agents who share a common interest in some data set. This implementation had a consumer include an instance of the DataSetSubscription class (which represents a subscription to some available data set) for each subscribed data set as a component of the description that is registered with the ADS. A search of the ADS can then be performed by creating a search template which includes an instance of DataSetSubscription representing the data set for which common interest is required. An ADS registered agent description will be returned as a result of the search if the equals function of DataSetSubscription returns true when used to compare the search template copy with the copy contained in the registered agent description.

The problem with this implementation is that each consumer must know the class definition of DataSetSubscription, which forces independent organisations to share code. A more suitable solution would make use of XMI and XPath querying: if an agent could register an XMI description of an object diagram for the data sets he is registered to, XPath querying (with the support of an ontology describing data set representations and subscriptions) could then be used to search for subscriptions of interest. The use of an ontology would remove the need for shared class definitions between organisations.

7.2 Update Strategies

It was identified in Section 6 that the lack of importance the CPS theory placed on the description of and relationships between the individual tasks involved in collaborative action, led to problems in the timing of task execution in the implemented framework. Update strategies in selective data replication require that consumers be aware of the other tasks being performed as part of the overall action, and execute any assigned task at a suitable time. As an example, in the case of scenario three the fire service cannot update the fire hydrant subset until the local proxy containing the shared replication has been initialised.

Extensive work should be put into the development of official update strategies for performing collaborative updates in an SDR community. This work should include how these strategies should be represented in order to capture all the required information such as the relationships between individual tasks. The current implementation uses the notion of collaboration contracts which works well, however the identifiers that are currently used in those contracts (fully qualified class names) should be altered to something which is implementation independent. An ontology has the potential to cover both the representation of tasks and the relationships between tasks. The contract identifiers could act as indexes into the ontology to determine the role that is being referenced.

In addition to the representation of task relationships in collaborative contracts and update strategies, the way in which these relationships are considered during collaborative action execution must be considered. One solution could be to force an agent to determine what task is executed directly after its own from the update strategy description, and find the peer who is responsible for executing that task and send an inform on the completion of the task signalling that the next task can now be executed.

7.3 Task Assignment

The current implementation of negotiating task assignment uses a contract-tender solution. This solution was chosen due to its success in overcoming a similar problem in a different domain. However this is not the only solution to negotiation and it may be advantageous to explore and evaluate the alternatives against the current solution.

One solution that could be attempted would be to implement some form of democratic negotiation system, where by consumers can vote for the member of the team which they would like the role to be assigned to. Such a solution would remove the influence that the team leader has over the task negotiation process.

7.4 Task Cost Computation

In order to tender for tasks of a collaborative contract, a consumer is required to determine and provide to the team leader a cost that will be incurred from executing these tasks. The cost of executing such tasks could be effected by a number of factors and determining what these factors are and the effect they have on task execution needs to be looked at more closely. The reasoning capabilities that are provided by the BDI architecture would allow a consumer to consider any number of factors in the determination of an overall cost.

7.5 Resource Management

Section 4.2.1 discussed what it meant for a consumer to have an interest in collaborating with a team to perform a replication update. The (un)availability of resources was identified as a reason for which a consumer may not be interested in collaborating. The effect of collaboration on the resources of an agent has not been considered in the framework implementation. For the purposes of this project consumers were created with only one goal: to perform updates collaboratively. In a commercial setting an agent may have many other goals and tasks to perform, joining a collaborative action may have a negative effect on the outcome of these. A study of the effect collaboration has on a consumers outside tasks could highlight the situations in which an agent should deny the chance to join a collaborative action.

7.6 Loss of Commitment

If a consumer chooses to become a member of some team, a responsibility to inform peers of an inability to complete a task that has been committed to is assumed. Section 5.3 identified team leader/member beliefs as the mechanism which allow a consumer to be aware of team membership. This ensures that the consumer is aware of the responsibility towards other team members, however when this responsibility the consequences of this responsibility become active remains a question.

An inability to complete a task that has been committed to could result in a number of situations. Future work is required to identify what may cause a consumer to come to the belief that a task can no longer be carried out. Having identified such situations it will be possible to ensure that the responsibility to communicate the inability to the team is upheld.

8 Conclusions

In this paper a framework for achieving collaborative updates in selective data replication has been proposed. The framework has been implemented using the OPAL agent platform and deployed within a group of agents posing as data consumers. The derivation of the framework has been based on the Cooperative Problem Solving theory which provides a high-level description of the tasks required from recognising the potential for collaboration right through to executing an action collaboratively. Scenarios for collaborative updates have been proposed and used for the identification of tasks required to be performed in collaborative action. These scenarios were used to determine the success of the framework in achieving the goal of collaborative updates in SDR.

The formalization of the model CPS provides resulted in a series of tasks which define a process that a group of agents can follow in order to complete each of the stages defined by CPS. The success of the formalization was proved through the implementation of the defined process using the OPAL agent platform, each stage of the CPS model can be identified in the execution of the framework implementation.

Execution of the collaborative update strategies defined by the proposed scenarios exposed an area where substantial future work will be required. CPS does not consider and makes no mention of the relationship between tasks that are performed as part of a collaborative action. In the case of SDR, these relationships have proved to be influential and the current representation of update strategies within the developed framework is not sufficient. Future work should focus on formally defining collaborative update strategies in SDR and the representation of these strategies within the framework developed here.

The work that has been performed in this project has exposed the size and complexity of creating a concrete implementation of software agent collaboration. A framework has been developed that goes some way towards achieving collaborative updates within SDR. The OPAL agent platform has proved useful in the implementation of the framework, in particular the micro-agent and role capabilities it provides. Areas of particular interest for future work have been identified, with experiences gained here providing a platform for what needs to come.

Bibliography

[1] Bratman, M. E., Israel, D. J., and Pollack, M. E. Plans and Resource-bounded Practical Reasoning. *Computational Intelligence* (1988).

[2] Cohen, P. R., and Levesque, H. J. Teamwork. *Nous, Special Issue on Cognitive Science and Artificial Intelligence* (1991).

[3] da Figueira Filho, C. S., and Ramalho, G. L. JEOPS - The Java Embedded Object Production System. *Advances in Artificial Intelligence: International Joint Conference, 7th Ibero-American Conference on AI, 15th Brazilian Symposium on AI* (2000).

[4] Foundation for Intelligent Physical Agents. *FIPA Agent Specification.* FIPA, 2000. Available from: `http://www.fipa.org/specifications/index.html`.

[5] Fox, S. Selective Data Replication using Intelligent Agents built on the OPAL Framework. *COSC366 Research Report* (2006 - 2007).

[6] Georgeff, M. P., and Rao, A. S. BDI Agents: From Threory to Practice. *Proceedings of the First International Conference on Multi-Agent Systems* (1995).

[7] Jennings, N. R. Commitments and Conventions: The Foundation of Coordination in Multi-Agent Systems. *The Knowledge Engineering Review* (1993).

[8] Khamis, A., Aboul-Ela, M., and Atwany, M. A Framework for Multi-agent Collaboration. In *Proceedings of Conference for Information and Computer Technology, ESISACT* (2004).

[9] Kinny, D., Ljungberg, M., Rao, A. S., Sonenberg, L., Tidhar, G., and Werner, E. Planned Team Activity. *MAAMAW '92: Selected papers from the 4th European Workshop on on Modelling Autonomous Agents in a Multi-Agent World, Artificial Social Systems* (1994).

[10] Kraus, S., Nirke, N., and Sycara, K. Reaching Agreements through Argumentation. In *Proceedings of the Twelfth International Workshop on Distributed Artificial Intelligence* (1993), pp. 233–247.

[11] Lochbaum, K. E., Grosz, B. J., and Sidner, C. L. Models of Plans to Support Communication: An Initial Report. *Proceedings of the Eighth National Conference on Artificial Intelligence* (1990).

[12] Lochbaum, K. E., and Sidner, C. L. Models of Plans to Support Communication: An Initial Report. In *Proceedings of the Eighth National Conference on Artificial Intelligence* (1990), AAAI Press, pp. 485–490.

[13] Pascoe, R. T., and Gu, X. Incorporating Update Semantics Within Geographical Ontologies. In *Proceedings of GeoSpatial Semantics* (2005), LNCS, pp. 211–226.

[14] Priestly, M. *Practical Object-Oriented Design with UML.* McGraw-Hill, 2000.

[15] Purvis, M., Cranefield, S., Nowostwaski, M., and Carter, D. OPAL: A Multi-Level Infrastructure for Agent-Oriented Software Development. *University of Otago, New Zealand* (2001).

[16] Sandholm, T. Automated Negotiation. *Commun. ACM 42*, 3 (1999), 84–85.

[17] Wang, M., Purvis, M., and Nowostawski, M. An Internal Architecture Incorporating Standard Reasoning Components and Standards-based Agent Communication. *International Conference on Intelligent Agent Technology* (2005).

[18] WILSKER, B. A Study of Multi-Agent Collaboration Theories. Available from: `citeseer.ist.psu.edu/wilsker96study.html`.

[19] WOOLDRIDGE, M., AND JENNINGS, N. R. Towards a Theory of Cooperative Problem Solving.

[20] WOOLDRIGE, M. The Belief-Desire-Intention Model of Agency. *Proceedings of the 5th International Workshop on Intelligent Agents V : Agent Theories, Architectures, and Languages* (1999).

A Example Execution Output

```
Police Consumer attempting to initiate collaborative update
Police Consumer found a common interest group of size: 2

Sent an ability request to: Ambulance Consumer

Sent an ability request to: Fire Consumer

Ambulance Consumer had ability requested by Police Consumer
for data set Roading provided by CCC

Fire Consumer had ability requested by Police Consumer
for data set Roading provided by CCC


Ambulance Consumer has ability to collaborate in strategies:
scenarioOne scenarioTwo scenarioThree

Fire Consumer has ability to collaborate in strategies:
scenarioOne scenarioTwo scenarioThree

Police Consumer received an ability inform from: Fire Consumer
Added to ability group: Fire Consumer

Police Consumer received an ability inform from: Ambulance Consumer
Added to ability group: Ambulance Consumer

Sent an interest request to: Ambulance Consumer

Sent an interest request to: Fire Consumer

Ambulance Consumer had interest requested by Police Consumer
for data set Roading

Fire Consumer had interest requested by Police Consumer
for data set Roading

Ambulance Consumer has interest

Fire Consumer has interest

Police Consumer received an interest inform from: Ambulance Consumer
Added to team: Ambulance Consumer
```

```
Police Consumer received an interest inform from: Fire Consumer
Added to team: Fire Consumer

Fire Consumer joined Team: -2892407622132463163 , led by Police Consumer

Ambulance Consumer joined Team: -2892407622132463163 , led by Police Consumer

Sent update contract to: Ambulance Consumer

Sent update contract to: Fire Consumer

Contract provided by leader of team -2892407622132463163
being evaluated by Fire Consumer

Contract provided by leader of team -2892407622132463163
being evaluated by Ambulance Consumer

Contract successfully evaluated by Fire Consumer
Contract successfully evaluated by Ambulance Consumer
Contract successfully evaluated by Police Consumer


Ambulance Consumer adding role
collaboration.scenarioOne.RetrievalRole to tender
Ambulance Consumer adding role
collaboration.scenarioOne.UpdateRole to tender

Fire Consumer adding role
collaboration.scenarioOne.UpdateRole to tender
Fire Consumer adding role
collaboration.scenarioOne.RetrievalRole to tender

Police Consumer adding role
collaboration.scenarioOne.UpdateRole to tender
Police Consumer adding role
collaboration.scenarioOne.RetrievalRole to tender

Ambulance Consumer tendered for role
collaboration.scenarioOne.UpdateRole at a cost of 0
Ambulance Consumer tendered for role
collaboration.scenarioOne.RetrievalRole at a cost of 13

Fire Consumer tendered for role
collaboration.scenarioOne.UpdateRole at a cost of 10
Fire Consumer tendered for role
collaboration.scenarioOne.RetrievalRole at a cost of 5

Police Consumer tendered for role
collaboration.scenarioOne.UpdateRole at a cost of 7
Police Consumer tendered for role
collaboration.scenarioOne.RetrievalRole at a cost of 9

Cheapest tender for role collaboration.scenarioOne.UpdateRole
submitted by:Ambulance Consumer
```

```
Cheapest tender for role collaboration.scenarioOne.RetrievalRole
submitted by:Fire Consumer

Fire Consumer about to attempt to
dynamically load role collaboration.scenarioOne.RetrievalRole
Police Consumer about to attempt to
dynamically load role collaboration.scenarioOne.RetrievalRole
Ambulance Consumer about to attempt to
dynamically load role collaboration.scenarioOne.UpdateRole

Police Consumer assumed role collaboration.scenarioOne.RetrievalRole
as requested by leader of team -2892407622132463163

Fire Consumer assumed role collaboration.scenarioOne.RetrievalRole
as requested by leader of team -2892407622132463163

Ambulance Consumer assumed role collaboration.scenarioOne.UpdateRole
as requested by leader of team -2892407622132463163

Ambulance Consumer informed the role collaboration.scenarioOne.UpdateRole
has been assumed and is ready for execution

Fire Consumer informed the role collaboration.scenarioOne.RetrievalRole
has been assumed and is ready for execution

Police Consumer informed the role collaboration.scenarioOne.RetrievalRole
has been assumed and is ready for execution

Ambulance Consumer executing role class collaboration.scenarioOne.UpdateRole
as part of team -2892407622132463163
Requesting dataset update from producer

Fire Consumer executing role class collaboration.scenarioOne.RetrievalRole
as part of team -2892407622132463163
Requesting retreival of dataset from UpdateRole

Police Consumer executing role class collaboration.scenarioOne.RetrievalRole
as part of team -2892407622132463163
Requesting retreival of dataset from UpdateRole
```