UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

Um Modelo Geral de Teamwork para Ambientes Dinâmicos com Requisitos de Tempo Real

Projeto de Pesquisa
Edital MCT/CNPq N^{o} 70/2008

Prof. Dr. Luis Otávio Campos Álvares
Proponente

Porto Alegre, dezembro de 2008

SUMÁRIO

RESUMO

Embora haja muitas pesquisas na área de *teamwork*, o desenvolvimento de times de agentes para ambientes complexos continua sendo um desafio, especialmente se esses ambientes possuem requisitos de tempo-real. Muitas ferramentas foram desenvolvidas, mas não existe uma solução geral, e as ferramentas mais gerais possuem sérios problemas com o requisito de tempo-real. Adicionalmente, a formação de times heterogêneos contendo agentes e humanos ainda apresenta algumas dificuldades, como a criação de um entendimento compartilhado entre humanos e agentes, e a previsibilidade mútua entre os membros do time. Assim, neste projeto pretende-se investigar e analisar os problemas de *teamwork* para ambientes dinâmicos com requisitos de tempo real para então propor novo modelo geral de *teamwork* que atenda a tal objetivo.

Palavras-chave: Teamwork, Sistemas Multiagentes, Inteligência Artificial.

1 APRESENTAÇÃO DO PPGC/UFRGS

O PPGC é um programa de pós-graduação em Computação do Instituto de Informática da UFRGS que desenvolve pesquisa com ênfase em 13 linhas de pesquisa principais. Avaliado pela CAPES com o conceito 6 desde 2006, tem 39 orientadores permanentes (25 bolsistas de produtividade do CNPq, sendo 14 nível 1 e 11 nível 2) e mais de 250 alunos de mestrado e doutorado. Tem participação constante nos principais eventos internacionais da área, projetos de cooperação com Universidades da Europa e Estados Unidos, além do reconhecimento do trabalho de grupos de pesquisa do Instituto de Informática no cenário científico nacional.

O PPGC oferece formação em Computação em dois níveis, Mestrado e Doutorado, e é voltado à formação de pesquisadores, docentes e profissionais qualificados para desenvolver atividades em empresas de alta tecnologia.

O ingresso no Mestrado em Ciência da Computação é anual e pressupõe o vínculo a um professor orientador desde o início do curso. A seleção é feita pela Comissão de Pós-Graduação com base no bom desempenho do candidato no POSCOMP, no seu curriculum vitae e nas suas cartas de recomendação.

O processo de seleção para o doutorado (para ingresso em fluxo-contínuo) envolve a análise da documentação do aluno e uma entrevista com o candidato. Há exigência de um plano de trabalho elaborado juntamente com o orientador, bem como experiência anterior do candidato em pesquisa. Com isso, procura-se garantir alunos com bom potencial para a pesquisa avançada.

O PPGC tem investido fortemente na formação de mestres e doutores em colaboração com instituições reconhecidas no exterior, incentivando e viabilizando a realização de estágios no exterior durante o doutoramento, bem como o doutoramento em co-tutela. Neste caso, ao final do curso, o aluno recebe o título de doutor pela UFRGS e também pela instituição estrangeira.

Em relação à interação com empresas, o PPGC está intensificando seus esforços no sentido de desenvolver o espírito de empreendedorismo em seus mestrandos e doutorandos, capacitando-os para atuar na indústria.

O Instituto de Informática mantém convênios e projetos de transferência tecnológica com diversas empresas, beneficiando diretamente o PPGC pela obtenção de recursos para laboratórios, bolsas e biblioteca. Em projetos apoiados pela Lei de Informática e/ou financiados pelos fundos setoriais, foram desenvolvidos projetos em cooperação com as empresas Altus, HP, Mimetic, Digitel, SOMAR, TechRobot, Dell, HP, CP Eletrônica, Digitel,

Digistar, Leader Tech, Teracom entre outros.

2 INTRODUÇÃO AO PROJETO

Teamwork[1] tem emergido como o paradigma para coordenação de agentes de forma cooperativa em ambientes dinâmicos e tem se mostrado capaz de levar a um comportamento robusto e flexível (SYCARA; SUKTHANKAR, 2006). Entretanto, não existe uma solução geral para construção de times de agentes e cada novo domínio pode ter novos requisitos. Uma classe de aplicação que se mantém como desafio para a área de *teamwork* é aquela na qual o ambiente é altamente dinâmico e que requer um tempo de resposta muito curto. O desafio aumenta quando o time pode ser composto por humanos, robôs e agentes artificiais.

Em ambientes dinâmicos, o estado do ambiente pode sofre alteração entre o agente perceber tal ambiente e atuar sobre ele. Ou seja, o tempo que o agente demora para agir pode afetar a qualidade de suas ações. Isso se torna mais grave quando o ambiente exige uma reatividade muito alta, com intensas atualizações de seu estado. Em domínios altamente dinâmicos a manutenção da coerência de time é também dificultada, dado que a mudança de postura do time precisa acompanhar as mudanças de estado do ambiente.

Aplicações como jogos de computador, podem representar um ambiente com tais características, onde a formação de *teamwork* pode necessitar uma abordagem diferenciada (MONTEIRO; ALVARES, 2008) (MONTEIRO; ALVARES, 2009). Nesses ambientes frequentemente é necessário sacrificar alguns aspectos gerais, como comunicação assíncrona ou decisões distribuídas, em favor do tempo de resposta. Assim, dependendo do domínio, ainda é possível formar times robustos e coerentes em ambientes altamente dinâmicos.

Teamwork tem sido aplicado a diversos domínios, como resgate em desastres (NAIR et al., 2000), combate militar (HILL et al., 1997; JONES et al., 1999), *robocup soccer* (KAMINKA, 1999; MARSELLA et al., 1999, 2001) e colaboração entre humanos e agentes (SCERRI et al., 2003). Um *teamwork* flexível, promovido por um modelo explícito, que define comprometimentos e responsabilidades para os membros do time, permite uma coordenação robusta, mantendo a coerência mesmo que haja falhas individuais e mudanças imprevisíveis no ambiente (STEVEN, 2005).

Algumas teorias foram propostas para formalizar o comportamento de um grupo de agentes para agir como um time. Duas dessas teorias possuem importantes resultados em problemas reais. Ela são *Joint Intentions* (LEVESQUE; COHEN; NUNES, 1990) e *Shared Plans* (GROSZ; KRAUS, 1996). Ambas são descritas através de um formalismo lógico,

[1] Um esforço cooperativo realizados por membros de uma equipe para alcançar um objetivo.

mas com diferentes abordagens. *Joint Intentions* foca na junção dos estados mentais do time, enquanto *Shared Plans* foca nas intenções do agente voltadas para acões de colaboração ou voltadas para ações conjuntas do grupo.

Baseadas nessas teorias surgiram ferramentas como *STEAM* (TAMBE, 1997a) e *Machinetta* (STEVEN, 2005) que tiveram bons resultados em diversos domínios de ambiente complexo (HILL et al., 1997; TAMBE; ZHANG, 1998, 2000; MARSELLA et al., 2001; JONES et al., 1999; PYNADATH; TAMBE, 2003; SCERRI et al., 2003; SCHURR et al., 2005; SCERRI et al., 2003, 2004; CHALUPSKY et al., 2001; SCERRI; PYNADATH; TAMBE, 2001, 2003; YEN et al., 2001). Entretanto, o *STEAM* foi desenvolvido usando uma versão antiga do *SOAR* (LAIRD; NEWELL; ROSENBLOOM, 1987), que teve mudança na sua linguagem desde então, e o *Machinetta*, que é um projeto derivado do *STEAM*, apresenta diversas limitações para ambientes altamente dinâmicos.

Com objetivo de superar as limitações de tempo de resposta das ferramentas anteriores, foi desenvolvido o *TWProxy* (MONTEIRO; ALVARES, 2008) (MONTEIRO; ALVARES, 2009), uma nova ferramenta que habilita agentes individuais a integrarem times e agir de forma coerente em ambientes altamente dinâmicos com requisitos de tempo-real. Ele utiliza algumas das boas idéias apresentadas pelo *Machinetta*, contornando algumas de suas limitações e adicionando novas características. Entretanto, o *TWProxy* não é genérico o suficiente para atender de forma adequada a times mistos de humanos e agentes artificiais pois não possui nenhum suporte a autonomia ajustável.

Nenhuma ferramenta atual oferece uma solução realmente geral para *teamwork*. A diversidade de domínios é o que, na prática, inviabiliza uma solução genérica que seja satisfatória para qualquer tipo de aplicação. Por isso, este projeto foca na investigação e análise de *teamwork* em ambiente multiagente altamente dinâmico, estocástico, parcialmente observável, com o objetivo de propor um modelo geral para esse tipo de ambiente, não atendido pelos modelos existentes.

3 CONTEXTO ATUAL

Diversos problemas multiagentes podem ser enquadrados como um problema de *teamwork*. Essencialmente, problemas de *teamwork* são aqueles onde um grupo de agentes compartilham um objetivo e precisam cooperar entre si agindo de forma coordenada para alcançar o objetivo. A diferença principal entre um problema de *teamwork* e de coordenação clássica é que em *teamwork* um objetivo compartilhado precisa ser alcançado com a cooperação dos agentes envolvidos. Para alguns domínios é possível ter um esquema de coordenação pré-combinado, assumindo a suposição da sala fechada(STONE; VELOSO, 1998), onde planos pré-definidos são executados pelo time após observar os ambientes e escolher o adequado. Em diversos ambientes dinâmicos, a suposição da sala fechada não é válida, o que torna a diferença entre *teamwork* e coordenação clássica ainda mais clara, quando é mais difícil manter a coerência de time. Abaixo são apresentados dois exemplos onde a coordenação clássica falha e é necessária a atuação em time.

3.1 Exemplo do comboio

Para ilustrar as diferenças entre *teamwork* e coordenação multiagente, Levesque (LEVESQUE; COHEN; NUNES, 1990) apresenta o exemplo de dirigir em comboio. Neste, existem dois agentes, *A* e *B*. O agente *B* deseja dirigir até sua casa mas não sabe o caminho, e *A* conhece o caminho para a casa de *B*. Assim, *A* pretende guiar *B* até um caminho que ele saiba seguir, cada um fazendo a sua parte. Neste exemplo, a coordenação clássica pode demonstrar vários pontos de falha de consistência.

Quando um agente acredita que seu objetivo ou foi alcançado, ou tornou-se impossível de alcançar, uma escolha racional seria abandonar o objetivo. Por exemplo, se o caminho que *A* conhecia está interditado e ele não conhece outro caminho que leve à casa de *B*, a intenção de guiar *B* deveria ser abortada. Neste caso, a coordenação não fornece nenhum compromisso entre *A* e *B*, não existindo nada que impeça que *A* simplesmente siga seu caminho. Se *B* percebe que conhece o caminho restante para sua casa, o objetivo para *B* foi satisfeito, então ele poderia seguir para casa sem considerar como *A* iria interpretar sua ação. Como *A* não tem consciência que o objetivo foi alcançado, continua tentando guiar *B* para o caminho de sua casa.

Até mesmo se um agente leva em consideração as ações do outro, a falta de coerência ainda poderia ocorrer. Se *A* faz um retorno, *B* poderia erroneamente concluir que *A* não está habilitado para acompanhá-lo e poderia executar outros planos para chegar em casa. Ou se *B* é forçado a parar, *A* poderia simplesmente interpretar que *B* sabe o caminho de

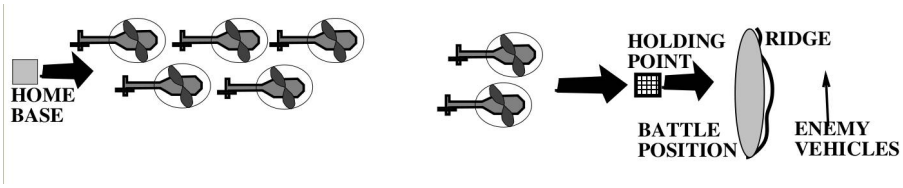

Figura 3.1: *Simulação militar de ataque de helicópteros(JONES et al., 1999).*

casa e continuar dirigindo sem ele.

3.2 Exemplo dos helicópteros militares

O exemplo dos helicópteros militares(JONES et al., 1999) envolve um caso de ataque como exemplificado na Figura 3.1. O grupo começa na base, onde o piloto comandante primeiro envia as ordens e instruções para os membros da equipe. O grupo processa as ordens e então segue até a posição de espera, próximo ao campo de batalha. Uma vez que o grupo alcança sua posição de espera especificada, ele aguarda enquanto os helicópteros de reconhecimento seguem ao campo de batalha. Baseados na comunicação entre os helicópteros de reconhecimento e o restante do grupo, os demais seguem para o campo de batalha para realizar um ataque. Uma vez que o ataque esteja completo, os helicópteros se reagrupam e voltam para base. Caso algum veículo inimigo seja encontrado pelo caminho em posição de ameaça, os demais são alertados para que seja feita uma nova rota que evite o veículo inimigo.

Nesse domínio haverá potenciais falhas de *teamwork* caso os membros da equipe não estejam comprometidos em manter a coerência do time. Abaixo são enumeradas possíveis falhas na coerência de time para esses dois domínios.

1. Após batalha com o inimigo, o comandante poderia retornar sozinho à base, abandonando os membros de sua equipe no campo de batalha.

2. Após a chegada do grupo no ponto de espera, um helicóptero de reconhecimento deveria se certificar de que a área atrás da colina estaria livre. Mas nessa verificação ele poderia ser abatido, deixando o resto da equipe esperando indefinidamente.

3. Um piloto, inesperadamente, processa as ordens iniciais antes dos outros. Ele então segue sozinho para o campo de batalha enquanto o restante do time continua na base.

4. Após helicópteros serem abatidos, apenas um de escolta chega na posição de espera. Este helicóptero espera indefinidamente os demais helicópteros para seguir em frente.

5. Quando a ordem inicial falha em atribuir o papel de escolta para um membro do time, o grupo inteiro espera indefinidamente quando chegar no ponto de espera.

6. As instruções enviadas pelo comandante para alguns membros foram perdidas, porque o comandante enviou a mensagem quando os membros da equipe estavam ocupados com outra tarefa. Neste caso, estes membros estão impossibilitados de selecionar as ações apropriadas.

7. Enquanto o grupo desvia de um veículo inimigo, inesperadamente um dos helicópteros destrói o veículo. Entretanto, os demais pilotos não são informados, assim uma rota alternativa é planejada desnecessariamente.
8. Em um caso extremo, quando a munição de todos os membros do time acaba, o grupo falha em inferir que a missão não poderia continuar.
9. Dois grupos de helicópteros acidentalmente usam o mesmo canal de rádio, levando à interferência e perda da mensagem inicial enviada pelo comandante. O grupo espera indefinidamente na base.

3.3 Modelos Teóricos de Teamwork

Trabalhos teóricos em *teamwork* (LEVESQUE; COHEN; NUNES, 1990) (GROSZ; KRAUS, 1996) (GROSZ; KRAUS, 1998) de agentes formalizam o comportamento de time e definem alguns pré-requisitos para a atuação coerente em equipe, tais como: os agentes precisam compartilhar os objetivos que eles querem alcançar, um plano de tudo que eles seguem juntos e conhecimentos do ambiente no qual eles operam; eles também necessitam compartilhar as intenções de executar um plano para alcançar um objetivo comum; os membros do time precisam ser conscientes das suas capacidades e como eles podem preencher os papéis necessários para os planos de alto nível do time; membros do time devem estar aptos para monitorar seus próprios progressos voltados aos objetivos do time, monitorar atividades de outros membros e as *Joint Intentions* do time(LEVESQUE; COHEN; NUNES, 1990). Muitos sistemas têm sido desenvolvidos usando essas idéias básicas de *teamwork*.

3.3.1 Joint Intentions

Segundo *Joint Intentions* (LEVESQUE; COHEN; NUNES, 1990), ações conjuntas de um time não consistem apenas de ações simultâneas e individualmente coordenadas. Para atuar em conjunto, um time deve ter consciência do estado da equipe e dos esforços do grupo como um todo. Por exemplo, no tráfego de automóveis, onde os motoristas atuam simultaneamente e são coordenados pelos sinais e leis de trânsito, não existe de fato *teamwork* envolvido. Mas se um grupo de motoristas decide fazer alguma coisa em conjunto, como por exemplo, dirigir em comboio, irá parecer que o grupo age como um único agente com crenças, objetivos e intenções próprias. Baseado nisso, *Joint Intentions* apresenta um modelo formal das propriedades mentais de um grupo de agentes que deseja executar ações conjuntas.

O formalismo de *Joint Intentions* foca na junção de estados mentais de membros de um time. Nele, para que um time faça uma ação em conjunto é necessário que seus membros se comprometam a completar a ação do grupo, enquanto mutuamente acreditam que é possível fazê-la. Além de se comprometer a realizar a tarefa, os membros de um time se comprometem a tornar de conhecimento comum o fato de um objetivo ser concluído, tornar-se impossível de se atingir ou deixar de ser importante.

Em *Joint Intentions*, assume-se que os agentes são modelados em um ambiente multiagente dinâmico, sem possuir uma crença completa, nem necessariamente correta, do mundo e dos demais agentes. Eles possuem objetivos modificáveis e suas ações são

falíveis. Por simplicidade é também assumido que uma vez o agente venha a pensar que o objetivo não é mais alcançável, ele nunca muda de idéia, e que sempre é possível alcançar a crença mútua.

3.3.1.1 Individual Commitment

O formalismo de *Joint Intention* é expresso em uma linguagem modal de primeira-ordem com o conectivo de igualdade e os operadores proposicionais. Nesta sub-seção são apresentadas as definições necessárias para compor as definições de comprometimento mútuo do time. A seguir são descritos os operadores da linguagem utilizada.

- $BEL(x, p)$, o agente x acredita em p.
- $GOAL(x, p)$, o agente x tem p como objetivo.
- $MB(x, y, p)$, os agentes x e y acreditam mutuamente em p.
- $AGT(x_1, \ldots, x_n, e)$, os agentes x_1, ..., x_n são os únicos agentes para a seqüência de eventos e.
- $e_1 \leq e_2$, e_1 é uma subseqüência anterior a e_2.
- $HAPPENED(a)$, a aconteceu.
- $HAPPENING(a)$, a está acontecendo.
- $HAPPENS(a)$, a irá acontecer.
- $a;b$, composição de ações.
- $a|b$, escolha não determinística.
- $a||b$, ocorrência concorrente de a e b.
- $p?$, teste de ação.
- a^*, repetição da ação a.

Dentro da modelagem é assumido que cada agente tem um conhecimento perfeito sobre suas crenças e objetivos, que as crenças são consistentes, e que os objetivos são consistentes entre si e com a crença.

Para simplificar a notação, são feitas as seguintes abreviações sintáticas paras as ações:

- $DONE(x_1, \ldots, x_n, a) \stackrel{def}{=} HAPPENED(a) \wedge AGT(x_1, \ldots, x_n, e)$
- $DOING(x_1, \ldots, x_n, a) \stackrel{def}{=} HAPPENING(a) \wedge AGT(x_1, \ldots, x_n, e)$
- $DOES(x_1, \ldots, x_n, a) \stackrel{def}{=} HAPPENS(a) \wedge AGT(x_1, \ldots, x_n, e)$

Os operadores modais $\diamond$ *eventually*, $\Box$ *always* e *until* são definidos a seguir com base nos predicados e operadores a cima:

- $\diamond p \stackrel{def}{=} \exists e HAPPENS(e; p?)$,
  p é verdade em algum ponto do futuro.

- $\Box p \stackrel{def}{=} \neg\Diamond\neg p$,
  p é sempre verdade a partir do momento atual.

- $UNTIL(p,q) \stackrel{def}{=} \forall c HAPPENS(c;\neg q?) \rightarrow \exists a(a \leq c) \wedge HAPPENS(a;p?)$,
  p é verdade até que q seja verdade.

Para concluir as definições de compromissos e intenções individuais, são definidos *PGOAL* e *INTEND*. *PGOAL* representando um objetivo individual persistente, onde o agente acredita que o seu objetivo ainda não foi atingido e que é possível atingí-lo. Assim, ele continua no objetivo enquanto ainda for possível ou relevante continuar. O *INTEND* representa a intenção do agente em executar uma ação, enquanto mantém um objetivo persistente que inclui tal ação e acredita que a está excutando.

- $PGOAL(x,p,q) \stackrel{def}{=} BEL(x,\neg p) \wedge GOAL(x,\Diamond p) \wedge UNTIL([BEL(x,p) \vee BEL(x,\Box\neg p) \vee BEL(x,\neg q)], GOAL(x,\Diamond p))$
  Onde x é o agente, p é o objetivo de x e q é o critério que torna p irrelevante.

- $INTEND(x,a,q) \stackrel{def}{=} PGOAL(x, DONE(x, UNTIL(DONE(x,a), BEL(x, DOING(x,a)))?;a), q)$,
  Onde x é o agente, a é a ação que o agente pretende fazer e q é o critério que torna irrelevante executar a ação.

3.3.1.2 Joint Commitment

Uma generalização do *PGOAL* para o caso onde o grupo atua como um agente é dada pelo predicado $JPG(x,y,p,q)$. Para se definir o *JPG* é necessário antes definir o objetivo mútuo MG e o objetivo fraco WG. No objetivo mútuo, os agentes envolvidos devem acreditar que todos possuem o mesmo objetivo e no objetivo fraco o agente envolvido continua com seu objetivo enquanto acredita que ele não foi alcançado, mas caso perceba que ele foi alcançado ou que se tornou inviável torna conhecimento mútuo o novo fato.

- $MG(x,y,p) \stackrel{def}{=} MB(x,y, GOAL(x,\Diamond p) \wedge GOAL(y,\Diamond p))$
  Onde x e y são agentes com um mesmo objetivo p

- $WG(x,y,p) \stackrel{def}{=} [\neg BEL(x,p) \wedge GOAL(x,\Diamond p)] \vee [BEL(x,p) \wedge GOAL(x, \Diamond MB(x,y,p))] \vee [BEL(x,\Box\neg p) \wedge GOAL(x,\Diamond MB(x,y,\Box\neg p))]$
  Onde x e y são agentes com um mesmo objetivo p

- $JPG(x,y,p,q) \stackrel{def}{=} MB(x,y,\neg p) \wedge MG(x,y,p) \wedge UNTIL([MB(x,y,p) \vee MB(x,y,p) \vee MB(x,y,\neg q)], MB(x,y, WG(x,y,p) \wedge WG(y,x,p)))$
  Onde x e y são agentes com o um objetivo p e com uma cláusula de relevância q

3.3.1.3 Joint Action

Dada a noção de comprometimento em conjunto é possível definir *Joint Intention JI* como uma generelização da intenção individual, que se resume em um comprometimento em conjunto para realizar uma ação enquanto todos, mutuamente, acreditam que estão executando a ação.

- $JI(x,y,a,q) \stackrel{def}{=} JPG(x,y,$
  $DONE(x,y,UNTIL(DONE(x,y,a),MB(x,y,DOING(x,y,a)))?;a),q)$
  Onde x e y são agentes envolvidos na ação em grupo, a é a ação executada em grupo e q a cláusula que torna a execução da ação a irrelevante.

3.3.2 Shared Plans

Em contraste com *Joint Intentions*, o conceito de *SharedPlans* (GROSZ; KRAUS, 1996) não é baseado na junção de atitudes mentais. Ao invés disso, *SharedPlans* remete ao conceito de atitudes intencionais, *intending-that*, o qual é similar a uma ação normal de um agente. Entretanto, um *intending-that* de um agente individual é voltado para ações de colaboração ou voltado a uma ação em conjunto do grupo. *Intending-that* é definido por um conjunto de axiomas que guia um indivíduo a realizar ações, incluindo ações de comunicação, que permitem membros do time, sub-time ou o time executar a tarefa atribuída(GROSZ; KRAUS, 1996).

Um *SharedPlans* pode ser representado por um *full SharedPlans*(FSP) ou por um *partial SharedPlans*(PSP). Um FSP para realizar uma ação α representa uma situação onde todos os aspectos de uma atividade conjunta α é completamente determinada. Isto inclui crença mútua e acordos em uma receita completa R_α para fazer a ação α. R_α é uma especificação de um conjunto de ações β_i, a qual, quando executada sobre as restrições especificadas contitui a execução de α. $FSP(P,GR,\alpha,T_p,T_\alpha,R_\alpha)$ denota o plano P de um grupo GR no tempo T_p para fazer a ação α no tempo T_α usando a receita R_α. De forma resumida, $FSP(P,GR,\alpha,T_p,T_\alpha,R_\alpha)$ é verdade se e somente se as seguintes condições são satisfeitas:

1. Todos os membros do grupo GR acreditam mutuamente que eles pretendem que a proposição $Do(GR,\alpha,T_\alpha)$ seja verdade, por exemplo, que GR faz α no tempo T_α.
2. Todos os membros de GR mutuamente acreditam que R_α é receita de α
3. Para cada passo de β_i em R_α:
   - Um sub-grupo $GR_k(GR_k \subseteq GR)$ tem um FSP para β_i, usando a receita $R_\beta i$. (GR_k pode ser apenas individual, no caso ele deve ter um *full individual plan*, uma analogia ao FSP só que individual.)
   - Outro membro de GR acredita que existe uma receita tal que GR_k pode levar a um β_i e ter um FSP para β_i (mas outros membros podem não conhecer $R_\beta i$)
   - Outros membros de GR pretendem que GR_k possa levar a β_i usando alguma receita.

A teoria de *SharedPlans* se propõe a descrever uma rede inteira de crenças e intenções de time quando engajado em *teamwork*. Nesta tentativa, um FSP representa um caso limite. Normalmente, quando engajado em uma atividade de time, um time só tem um *partial SharedPlan*(PSP). O PSP é uma foto do estado mental do time em uma situação particular em seu *teamwork*, e comunicação e planejamento são freqüentemente utilizados para completar as condições de um FSP(Embora, em um ambiente dinâmico, o time possa nunca realmente formar um FSP). Abaixo seguem três exemplos nos quais podem existir um PSP.

- A receita R_α pode ser apenas parcialmente especificada. Como o time decide reati-

vamente o próximo passo baseado no contexto e na situação atual, poderiam ser consideradas evoluções das receitas ao longo do tempo, principalmente em ambientes dinâmicos. De acordo com a teoria de *SharedPlans*, membros do time devem chegar a uma crença mútua nos próximos passos β_i. Para cada passo β_i na receita, o subgrupo relevante deve formar um *SharedPlan*.

- A alocação de tarefas do time pode não ser conciliada. Por exemplo, o agente ou o grupo para executar uma tarefa pode não ser determinado. Nesta situação, membros do time pretendem que exista algum indivíduo ou sub-grupo para fazer a tarefa. Dentre as ações consideradas como um resultado de um *intending-that*, indivíduos podem se voluntariar para executar a tarefa não conciliada ou persuadir/ordenar outros a pegar a tarefa.

- Indivíduos ou sub-grupos podem não ter atingido uma crença mútua apropriada para formação de um FSP, levando à comunicação dentro do time. Essa comunicação também pode surgir devido às atitudes de *intenting-that* dos agentes, ambas voltadas aos objetivos do time e às atividades dos membros do time. Por exemplo, um membro do time pretende que o time faça uma ação β_i, e acredita que a comunicação de alguma informação particular irá habilitar o time a fazer β_i. Isso irá levá-lo a comunicar tal informação ao time, desde que a comunicação não conflite com os compromissos anteriores.

3.4 Ferramentas de Teamwork

Baseadas em *Joint Intentions* e *SharedPlans* surgiram várias ferramentas que auxiliam na manutenção da coerência de *teamwork*, tais como: *GRATE** (JENNINGS, 1995), *COLLAGEN* (RICH; SIDNER, 1997), *STEAM* (TAMBE, 1997a), *RETSINA* (SYCARA et al., 2001), *MONAD* (VU et al., 2003), *Machinetta* (SCERRI et al., 2004) e *TWProxy* (MONTEIRO; ALVARES, 2008) (MONTEIRO; ALVARES, 2009).

*GRATE** é uma versão extendida do *GRATE*(*Generic Rules and Agent model Testbed Environment*), um sistema integrado de propósito geral que contém conhecimento genérico sobre cooperação embutido. O *GRATE** é baseado em *Joint Responsibility*, que por sua vez é baseado em *Joint Intentions*. Ele propõe uma arquitetura integrada de agente com duas camadas, uma do sistema de domínio e outra de controle e cooperação, tendo a sua principal aplicação no domínio da indústria.

O *COLLAGEN* é uma ferramenta feita em *Common Lisp* e baseado em *SharedPlan* que incorpora a idéia do discurso colaborativo(RICH; SIDNER, 1997). Ele foi inicialmente desenvolvido para uma aplicação simples de escolha de rotas aéreas e mantém o foco na interação entre humano e agentes. O *COLLAGEN* segue o paradigma de que a colaboração de agentes com humanos deve ser governada pelos mesmos princípios que seguem as interações entre as pessoas.

RETSINA é uma infraestrutura de sistemas multiagentes que utiliza mecanismo de raciocínio de time baseado em *SharedPlans*, com os objetivos de traçar tarefas interdependentes para membros do time, reconhecer e reportar conflitos, propor soluções para resolver os conflitos e monitorar a performance do time.

O *STEAM* é um modelo genérico para implementação de *teamwork*. Foi desenvolvido

sobre a plataforma *SOAR*(LAIRD; NEWELL; ROSENBLOOM, 1987) e é baseado nas teorias de *Joint Intentions* e *SharedPlans*. Diversos resultados demonstram o seu sucesso na composição de times de agentes em ambientes dinâmicos (TAMBE, 1997b,a; HILL et al., 1997; TAMBE; ZHANG, 1998; JONES et al., 1999; TAMBE; ZHANG, 2000; MARSELLA et al., 2001).

O *MONAD* (VU et al., 2003) define toda uma arquitetura de controle multiagente inspirada no funcionamento do *STEAM*. Ele integra o projeto do time baseado em programação de *script off-line* com um motor de coordenação em tempo de execução. Seu principal avanço em relação ao *STEAM* é permitir que o projetista defina diferentes metodos de arbitragem multiagente, além de fornecer um conjunto de ferramentas gráficas para do desenvolvimento de sistemas multiagentes.

Machinetta é uma evolução do modelo de *teamwork* do *STEAM*. Implementado em Java, ele segue uma abordagem baseada em *proxy*, que visa habilitar agentes não comprometidos socialmente ao comportamento social. Introduz um novo algoritmo de alocação de papéis e tem como característica importante a cooperação entre robôs, agentes e humanos através do ajuste de autonomia das entidades.

Devido a limitações das ferramentas existentes em fornecer suporte de *teamwork* para ambientes altamente dinâmicos, foi desenvolvido o *TWProxy*. Baseado em *Joint Intentions*, o *TWProxy* utiliza algumas características do *Machinetta*, superando suas principais limitações em atender a requisitos de tempo-real e introduzindo novas características úteis para domínios altamente dinâmicos. A seguir, são descritas algumas dessas ferramentas citadas.

3.4.1 Infraestrutura RETSINA

RETSINA é uma infraestrutura aberta de sistema multiagente que suporta comunidades de agentes heterogêneos. O sistema *RETSINA* foi implementado com a idéia de que o agente deveria formar uma comunidade de pontos que se engajam em uma relação ponto a ponto. Qualquer estrutura de coordenação na comunidade de agentes deve emergir da relação entre os agentes ao invés de imposta pela infraestrutura. Seguindo essa premissa, *RETSINA* não emprega controle centralizado em sistemas multiagentes, em vez disso ele implementa serviços de infraestrutura distribuída que facilitam a relação entre agentes em vez de gerenciá-los.

O *RETSINA* é definido como um conjunto de serviços, convenções e conhecimentos que dá suporte às interações sociais complexas. Os agentes precisam dos serviços para tornarem-se aptos a encontrar os outros em um ambiente aberto, para se comunicar e para garantir que as restrições de seguranças serão satisfeitas. Convenções, como linguagem de comunicação de agentes e políticas de conversação são a base para alcançar o acordo do que os agentes estão fazendo e o que eles estão alcançando. O agente precisa também do conhecimento de como usar a infraestrutura, as linguagens de comunicação e protocolos, bem como uma ontologia comum para ter uma participação efetiva na comunidade.

A organização da infraestrutura de sistemas multiagentes *RETSINA* é mostrada na Figura 3.2. Ela apresenta como os vários componentes são organizados. O restante dessa seção descreve cada um desses componentes.

- **Ambiente de Operação:** O *RETSINA* é independente da plataforma na qual os

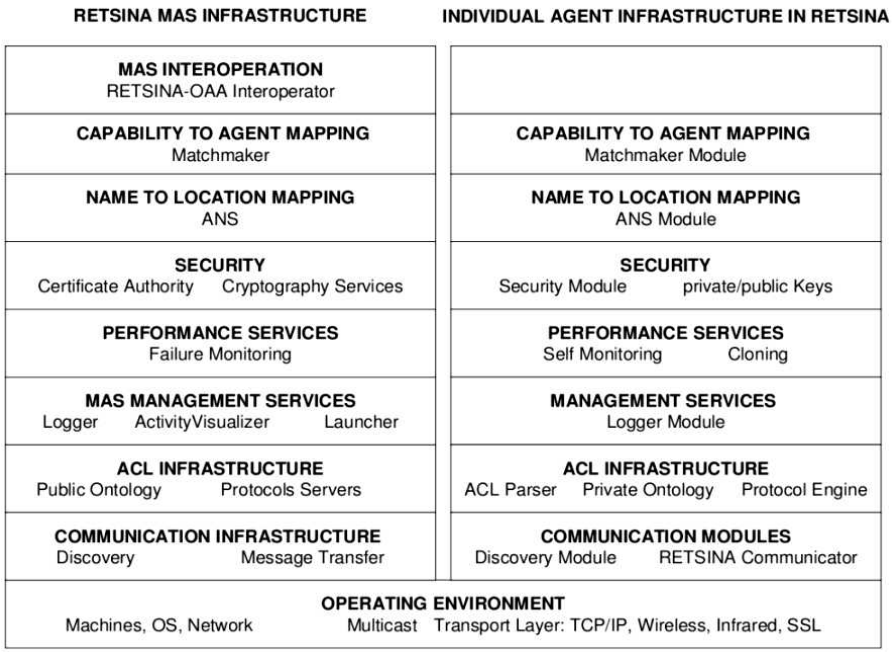

Figura 3.2: *Infraestruta multiagente RETSINA e infraestrutura de agente individual (SYCARA et al., 2001).*

componentes de infraestrutura e os agentes executam. Ele automaticamente lida com os diferentes tipos de camadas de transporte.

- **Infraestrutura de comunicação:** O *RETSINA* é baseado em dois tipos de canais de comunicação. Um provê transferência de mensagens por comunicação direta ponto a ponto entre os agentes, a outra é baseada em *multicast* usada para descobrir processos que levem o agente a achar os componentes de infraestrutura.

- **Infraestrutura de linguagem de comunicação de agente:** A troca de mensagens entre agentes no RETSINA é feita em KQML(FININ et al., 1994). As mensagens possuem dois componentes: a especificação do conteúdo da mensagem e o envelope da mensagem. No envelope são especificadas informações como o remetente, o destinatário, o histórico da conversação, a ontologia e a linguagem utilizadas no conteúdo. O formato do envelope é fixado pelo *RETSINA*, entretanto nenhuma suposição é feita quanto ao conteúdo da mensagem.

- **Gestão de serviços:** O gerenciamento de aplicações é uma tarefa complexa que fica mais difícil com o tamanho da aplicação e o aumento do número de máquinas e agentes. Ele inclui três componentes de gerenciamento: o *Logger*, *Activity Visualizer* e *Laucher*, que formam um conjunto de ferramentas para ajudar a depurar, monitorar e lançar alguma aplicação.

- **Serviços de execução:** O *RETSINA* provê serviços de monitoramento na simulação, *check-pointing* e *roll-back* distribuídos.

- **Segurança:** Como o *RETSINA* é um sistema aberto, agentes desconhecidos e possivelmente não confiáveis podem entrar a qualquer momento. Esses agentes podem causar danos ao sistema de várias formas: eles podem espionar outros agentes, roubar informações e danificar o conteúdo dos componentes de infraestrutura. A infraestrutura de segurança do *RETSINA* entretanto, previne que tais problemas

aconteçam.

- **Mapeamento de nome para localização:** Este componente provê um significado à localização física do agente através do mapeando em um ID. Assim, quando um agente precisa saber o endereço de um outro agente, ele pergunta para o componente de mapeamento.

3.4.2 STEAM Framework

O *STEAM*(TAMBE, 1997a) é um modelo geral de *teamwork* construído com base nos trabalhos teóricos de *Joint Intentions*(LEVESQUE; COHEN; NUNES, 1990) e de *SharedPlans*. Ele foi desenvolvido com o objetivo de contornar problemas envolvidos no desenvolvimento de times de agentes para ambientes complexos. Isto porque, até então os sistemas multiagentes implementados freqüentemente falhavam em fornecer a flexibilidade necessária na coordenação e comunicação para manter a coerência do time em tais domínios. A maioria desses sistemas era suprida com planos pré-definidos e coordenação específica para o domínio. Quando colocadas em um domínio dinâmico com incertezas, a inflexibilidade da coordenação pré-planejada leva a grandes falhas, pois em muitos casos é praticamente impossível pré-planejar todas as situações possíveis. Com o crescimento da complexidade das situações de *teamwork*, essas abordagens aumentavam os casos de falha. Outro problema existente era o fato da coordenação estar ligada a domínios específicos, o que inviabiliza o reuso em outros domínios.

Assim, o *STEAM* surge para prover aos agentes um modelo geral de *teamwork* que possibilita contornar essas dificuldades. Tal modelo permite ao agente raciocinar de forma autônoma sobre a coordenação e a comunicação, provendo os requisitos de flexibilidade no *teamwork*. Este modelo genérico também permite o reuso da capacidade de *teamwork* em diversos domínios. Tal reuso não apenas economiza tempo de implementação, como também garante a consistência do *teamwork* através das aplicações.

No *STEAM*, *Joint Intentions* são usadas como blocos de construção do *teamwork*. Algumas vantagens são tiradas disso. A primeira é que o comprometimento em *Joint Intention* provê o princípio para o raciocínio sobre a coordenação e a comunicação em *teamwork*. Segunda, a junção de comprometimentos em *Joint Intentions* provê o guia para monitorar e manter as atividades do time. Terceiro, um *Joint Intention* leva para uma representação explícita de uma atividade de time, e assim facilita o raciocínio sobre *teamwork*.

Adicionalmente, para garantir a coerência do time, o *STEAM* utiliza o *SharedPlans* para resolver questões como as seguintes:

- Evitar que um agente interfira de forma destrutiva nas ações de outros.
- Controlar a quantidade de informação compartilhada.
- Replanejar quando o objetivo é visto como inalcançável.
- Generalizar a capacidade de comunicação.

Membros do time devem possuir um caminho de solução comum em seus *Joint Intentions* para um objetivo de alto-nível. Sem essa restrição, membros do time poderiam possuir soluções alternativas que cancelam uns aos outros, não gerando progresso em direção ao objetivo. A teoria de *SharedPlans* resolve tal coerência seguindo o *intentions-that*

voltado ao objetivo. A abordagem do *STEAM* é paralela com *SharedPlans*, entretanto ele é construído sobre *Joint Intentions*. O *STEAM* usa *Joint Intentions* como bloco de construção para hierarquicamente construir um estado mental de atitudes do membro do time individual e garantir que os membros do time possuam um caminho de solução comum. Em domínios dinâmicos, dado planos reativos, uma receita R_α pode evoluir passo a passo durante a execução. No *STEAM*, como a receita evolui, se o passo β_i requer execução por todo o time, o *STEAM* requisita que todo o time concorde com β_i e forme um *Joint Intention* para executá-lo. Para executar um sub-passo de β_i outros *Joint Intentions* são formados, levando a uma hierarquia. Durante a expansão da hierarquia, se um passo envolve apenas um sub-time então este sub-time deve formar um *Joint Intentions* para executar esse passo. Se só um indivíduo está envolvido no passo, este forma uma intenção de fazer o passo. Em geral, o resultado da hierarquia de intenções evolui dinamicamente, dependendo da situação em que o time se encontra.

SharedPlans também ajuda a resolver questões como a quantidade de informação que os membros do time devem manter sobre os outros, particularmente quando o passo β_i envolve apenas um sub-time ou um indivíduo. O *STEAM* requer que caso o passo β_i seja executado por um sub-time, membros do time mantenham um rastro das *Joint Intentions* do sub-time para executar o passo. Este rastro intencional não precisa envolver detalhes do plano. Um membro do time deve apenas ser apto a inferir o que os outros membros pretendem fazer no passo β_i.

O *STEAM* precisa formar um *Joint Intention* para replanejar sempre que um objetivo de time para executar β_i é visto como inalcançável. O replanejamento pode levar o time primeiro a analisar a causa da não alcançabilidade. Dentre outras possibilidades, a causa poderia ser a ausência de uma sub-tarefa para o sub-time ou indivíduo, ou a falha de um sub-time ou indivíduo na execução de uma tarefa relevante. Em tal caso, cada membro do time atua para determinar o agente apropriado ou o sub-time para executar a tarefa relevante. Como resultado, um agente pode se voluntariar, ou sugerir outro indivíduo ou sub-time para executar a tarefa.

A capacidade de comunicação do *STEAM* é generalizada via abordagem híbrida que combina *Joint Intention* como aspectos de *SharedPlans*. A comunicação de *Joint Intention* poderia potencialmente ser alcançada em *SharedPlans* através de axiomas definidos como *intention-that*. Como por exemplo, considerando que um membro do time obteve informações privadas sobre o sucesso da ação β_1 do time. Em *Joint Intentions* esse membro do time tentará uma crença mútua de que β_1 teve sucesso, levando à comunicação. Em contraste, em *SharedPlans*, a comunicação de membros do time deveria aparecer porque: uma ação β_2 segue β_1, e o time não pode fazer β_2 sem todos tomarem conhecimento que β_1 foi feito com sucesso. Assim, baseado na interrelação entre as ações, é possível raciocinar e derivar a comunicação em *SharedPlans*, mas *Joint Intentions* provê tal comunicação sem a necessidade de raciocínio.

Assim, o *STEAM* inicia com *Joint Intentions*, mas quando constroí a estrutura hierárquica ele tem um paralelo com a teoria de *SharedPlans*, em particular, *Partial SharedPlans*. O resultado pode ser considerado como modelo híbrido, que une a formalização do comprometimento de *Joint Intentions* e o tratamento detalhado das atitudes do time em tarefas complexas do *SharedPlans*.

A base do *STEAM* é a execução hierárquica de planos reativos. Estes planos reativos são implementados com operadores em *SOAR*(LEWIS, 1999). O *STEAM* introduz a idéia de

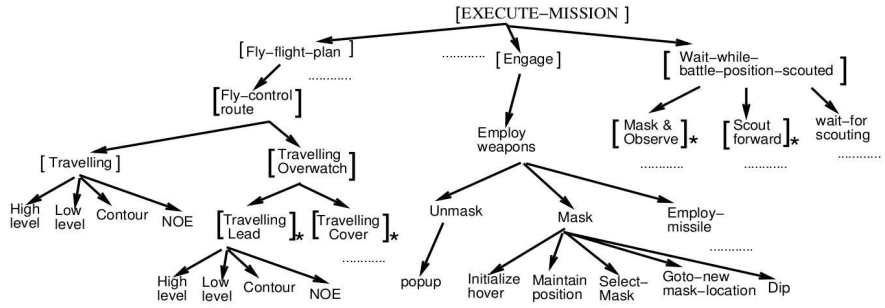

Figura 3.3: *Hierarquia de operadores para o domínio de ataque aéreo(TAMBE, 1997a)*

operador de time, que quando instanciado por um agente, forma uma *Joint Intention* do time. Os operadores de time expressam explicitamente uma junção de atividades de time, diferente do operador individual regular, que expressa a atividade do próprio agente. Na hierarquia apresentada na figura **??**, os operadores de time são envolvidos em colchetes, enquanto os outros são operadores individuais.

Como os operadores individuais, os operadores de time também consistem de regras de precondições; regras de aplicação; e regras de terminação. Dado um operador *OP* arbitrário, todos os membros do time devem simultaneamente selecionar *OP* para estabelecer um *Joint Intention*. Na Figura **??**, em alto nível, o time forma um *Joint Intention* para $[execute - mission]_{\theta}$, na execução desse *Joint Intention*, o time pode fomar o `Joint Intention` $[engage]_{\theta}$ e para executar $[engage]_{\theta}$, todos os membros do time devem selecionar individualmente os operadores $employ - weapongs$ formando intenções individuais. Assim, uma hierarquia inteira de intenções individuais e conjuntas é formada quando um agente participa de um trabalho em grupo.

Apesar de representar um marco no desenvolvimento de *teamwork*, a utilização do *STEAM* em sistemas atuais é dificultada devido a existência de complicadores. Um deles é o sistema baseado em regras (*SOAR*) sobre o qual o *STEAM* foi desenvolvido. Como o *STEAM* é representado por um conjunto de operadores e regras genéricas de time na linguagem do *SOAR 7*, mudanças na linguagem utilizada por versões mais recentes do *SOAR* impõe a necessidade reescrever algumas das regras e operadores do *STEAM*, o que pode inviabilizar a sua utilização. O próprio grupo desenvolvedor do *STEAM* fala do *Machinetta* como uma evolução do *STEAM*.

3.4.3 Machinetta Proxy

O *Machinetta* é uma ferramenta para o desenvolvimento de time de agentes que derivou de trabalhos como o *STEAM* e o *TEAMCORE* (TAMBE et al., ????). Ele usa uma abordagem de *proxy*, ou seja, faz um papel de intermediador de time para agentes heterogêneos, podendo formar inclusive time contendo agentes artificiais, humanos e robôs. Uma das suas principais inovações frente aos seus antecessores é a autonomia ajustável, que permite, por exemplo, que humanos assumam papéis que estejam sendo executados por agentes. O *Machinetta* também introduz um novo algoritmo de alocação de papéis, que em teoria permite a formação de times com grande quantidade de agentes.

Um padrão que tem emergido para a criação de times robustos e altamente heterogêneos

é a arquitetura que utiliza *proxies* semi-autonômos para criar um camada homogênea de coordenação para agente altamente heterogêneos (TAMBE et al., ????). Provendo cada agente com um *proxy* que tem o conhecimento de time, a habilidade de coordenação de cada par agente-*proxy* é uniformizada.

O *proxy* gerencia a coordenação e executa operações de rotina que são necessárias para a cooperação, como: informar aos demais quando um plano já foi completado, não pode mais ser completado ou tornou-se irrelevante; achar RAPs (*Robot-Agent-Person*) para preencher um determinado papel; quando necessário, ajustar o plano que o grupo está seguindo; e compartilhar informação para garantir a operação continua do time, enquanto deixa os RAPs livres para suas atividades. Nessa abordagem, a coordenação do time é alcançada para cada domínio dando ao membro do time um *proxy* que assume as rotinas de coordenação em nome do RAP membro do time (PYNADATH; TAMBE, 2003).

Assim, um time de *proxies* implementa um Programa Orientado a Time (POT). Um POT é uma descrição das atividades que precisam ser executadas em nível de time para alcançar o objetivo. Ele consiste de planos de time reativos, papéis, relacionamento entre papéis, e condições para iniciar e terminar um plano. Os *proxies* instanciam os planos dinamicamente quando, durante a execução, o seu estado atual casa com as condições requeridas para lançar o plano. O controle de comunicação do *proxy* determina quais mensagens devem ser enviadas dentre os proxies para garantir que seja mantida a coesão.

Com *proxies* homogêneos é possível escrever um POT que será executado de acordo com o algoritmo de coordenação utilizado pelo *proxy*. Na criação dos POTs os programadores não precisam se preocupar com detalhes específicos de baixo nível da coordenação. Isto porque eles especificam as atividades do time utilizando apenas primitivas de alto nível. Escrevendo POTs neste nível de abstração fica mais fácil para o programador especificar rapidamente atividades complexas de time.

Os planos de time provêm uma representação explícita dos objetivos do time que são mantidos conjuntamente por seus membros. Eles permitem que membros do time simplifiquem seus raciocínios e concentrem-se apenas em suas tarefas que são diretamente relevantes para o atual objetivo do time. Assim, eles dão forma ao comportamento do time, usando comportamentos individuais como meios para alcançar um objetivo comum.

Toda a coordenação no *Machinetta* é feita através da alocação de papéis. Os planos de alto nível definidos em um POT determinam quais são as possíveis ações que um time pode executar. Essas tarefas são representadas através de papéis que são atribuidos quando um novo plano é lançado.

Uma contribuição importante do *Machinetta* é seu algoritmo de alocação de papéis, o *LA-DCOP*. Esse algoritmo utiliza uma abordagem de *token*, no qual um *token*, que representa um papel a ser alocado, é passado entre os RAPs até que algum deles aceite ficar com o papel. Assim, a decisão de quem fica com o papel é feita através da verificação de um limiar que o *token* carrega, onde aquele que possui habilidade superior a aquela especificada no limiar pré-estabelecido do *token*, fica com o papel.

O *software* do *Machinetta proxy* é formado por cinco componentes, como visto na Figura **??**.

- Comunicação: Trata a comunicação com outros *proxies*.
- Coordination: Trata o raciocínio sobre planos de time e comunicação.

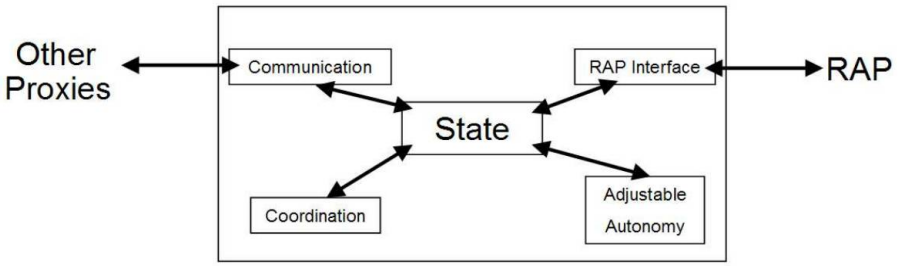

Figura 3.4: *Arquitetura do Machinetta Proxy (SCERRI et al., 2004).*

- Estado: A memória de trabalho do *proxy*.
- Autonomia Ajustável: Raciocina sobre quando atuar de forma autônoma ou passar o controle pra outro membro do time.
- Interface RAP: Trata a comunicação com os membros do time.

Os *proxies* por si só não possuem a habilidade de alcançar objetivos no nível do domínio. Ao invés disso, eles precisam garantir que todas as necessidades requisitadas em nível de domínio são atendidas através da instanciação de papéis apropriados atribuidos aos RAPs devidos.

Toda a execução do *proxy* é dirigida a mensagem. Quando um mensagem chega de um RAP ou de outro *proxy*, uma nova crença é adicionada ao estado do *proxy*. As crenças no estado constituem o conhecimento do *proxy* da situação do time e do ambiente. Este estado opera como um quadro-negro, com componentes escrevendo informações no quadro-negro e outros reagindo a informação escritas.

Para o *Machinetta*, criar o entendimento compartilhado entre humanos e agentes membros de um time é o maior desafio face ao desenvolvimento de iniciativas mistas com organizações de humanos e agentes. O componente de autonomia ajustável resolve as circunstâncias sobre quando o *proxy* deveria agir de forma autônoma ou quando deveria esperar por entrada de um membro do time. Tal raciocínio é vital para o sucesso da implantação de times heterogêneos contendo pessoas. Entretanto, os outros componentes do *proxy* são isolados do processo de raciocínio pelo componente de autonomia ajustável, assim esses componentes precisam conhecer apenas se a última decisão feita pelo *proxy* foi feita de forma autônoma ou por um membro do time.

Através da autonomia ajustável, um agente pode dinamicamente variar o nível de autonomia com o qual ele atua, permitindo explorar habilidade humanas para melhorar sua performance de tarefas, sem tornar-se necessariamente dependente de uma interação humana. Por este motivo, a autonomia ajustável é vista como um ponto crítico para o sucesso de um sistema multiagente com suporte a interação humana.

A formação de times com humanos e agentes artificiais carrega alguns desafios que vão além da interação individual entre humano e agente artificial. O time pode ter sua coordenação afetada enquanto espera uma resposta humana, um time de agentes artificiais pode potencialmente tornar globais decisões que são inaceitáveis para times de humanos, e a diversidade da organização humana no mundo real pode necessitar que agentes gradualmente aprendam modelos individualizados dos humanos membros do time, tomando decisões razoáveis mesmo antes de ter informações suficientes disponíveis.

No *Machinetta*, o componente de interface RAP é o principal ponto de extensão do *proxy* para integração com um novo RAP. Nesta interface é tratada a forma como o RAP e o *proxy* compartilham informação. Por exemplo, a interface RAP para um bombeiro que está atendendo uma situação de desastre poderia ser uma grande interface gráfica, enquanto para um agente artificial bastaria uma comunicação simples via *socket*.

3.4.4 TWProxy

Apesar de existirem soluções de *teamwork* para ambientes dinâmicos, não existia, por parte das ferramentas anteriores ao *TWProxy* (MONTEIRO; ALVARES, 2008) (MONTEIRO; ALVARES, 2009), uma preocupação evidente com o tempo de resposta no processo de mudança de atividades dentro do time a ponto de atender a requisitos de tempo real. Essa característica é muito importante quando o ambiente é altamente dinâmico e a qualidade das ações do time pode cair muito com o atraso na mudança de uma estratégia. Por isso, o *TWProxy* mantém o foco na diminuição do tempo de resposta, sem perder a qualidade do gerenciamento das ações de time.

O *TWProxy* é uma nova infraestrutura leve e eficiente que promove coordenação para times de agentes heterogêneos, com a finalidade de atender aos requesitos para o desenvolvimento de times de agentes em ambientes dinâmicos de tempo-real. *TWProxy* é baseado no formalismo de *Joint Intentions* e inspirado no *Machinetta*, superando algumas de suas limitações e adicionando novas características importantes como:

- uma linguagem simples e extensível para descrição de crenças e planos,
- dois processos eficientes de alocação de tarefas parcialmente distribuídos,
- manutenção de consistência através de comunicação atômica,
- reuso de planos terminados,
- planos invariantes ao número de agentes.

Na abordagem apresentada pelo *TWProxy*, cada agente é associado a um *proxy*, como visto na Figura 3.5. O *proxy* fornece ao agente uma interface de time, permitindo que ele participe de uma equipe sem se preocupar com os detalhes de coordenação. Tudo que ele precisa para compor um time é agir conforme as sugestões do *proxy* e informar a seu *proxy* sobre a atualização de determinadas crenças. Assim, os *proxies* ficam responsáveis por manter a coerência de time através da comunicação *inter-proxy* e do planejamento realizado pelo grupo de *proxies*. O acoplamento simbiôntico do agente individual com o seu *proxy* é o que forma o agente social, onde toda a comunicação e planejamento de time são feitos através dos *proxies*, sem necessidade de consciência social do agente individual.

A forma utilizada pelo *TWProxy* para alcançar a coordenação multiagente é através da alocação de papéis, que pode ser vista também como a alocação de tarefas ou atividades de time. Essa maneira de coordenar os agentes é flexível o suficiente para permitir que agentes heterogêneos componham a equipe, já que o *proxy* não diz como o agente deve executar as atividades. O conhecimento de como executar a tarefa fica por conta do próprio agente e este deve ter conhecimento do que ele está capacitado para fazer. Assim, quando ele informar ao *TWProxy* sobre suas capacidades estará habilitando o *proxy* a realizar um alocação adequada. Como os *proxies* comunicam-se entre si, cada *proxy* conhece as habilidades dos demais membros da equipe.

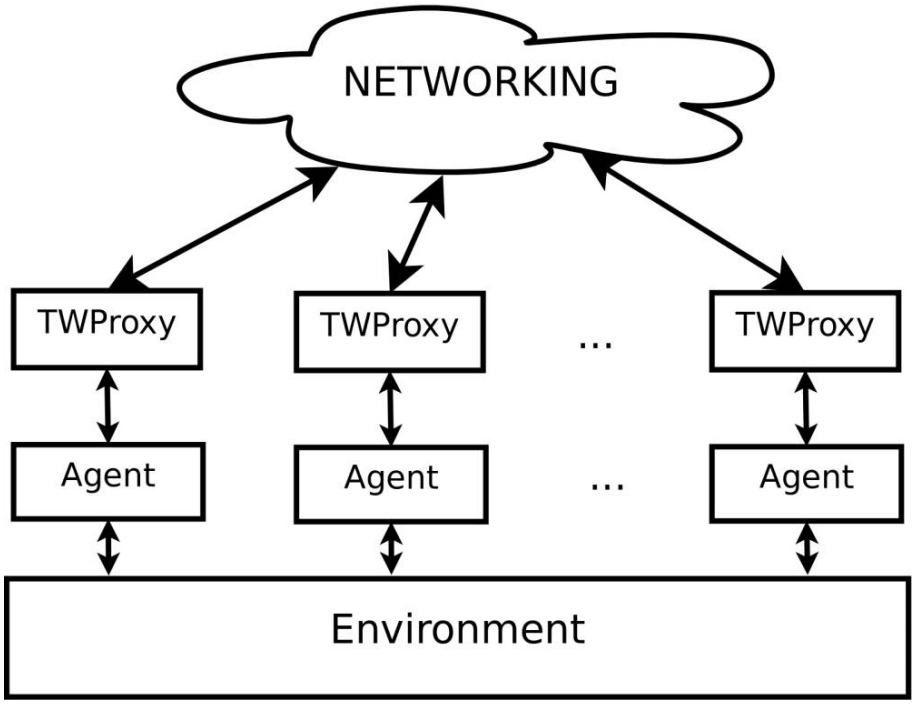

Figura 3.5: *Teamwork model based on proxy.*

TWProxy também provê a funcionalidade de quadro-negro distribuído, pois o mesmo mantém o conhecimento compartilhado consistente. Este repositório de crenças pode ajudar os agentes a tomarem decisões. Como o ambiente é parcialmente observável e as percepções dos agentes possuem restrições, o conhecimento compartilhado pelo *TWProxy* reflete um conjunto de crenças que não poderiam ser diretamente percebidas por um único agente.

A Figura 3.6 mostra a organização interna do *TWProxy*. A seta 1 indica a chegada de uma nova crença percebida pelo agente que irá atualizar a base de conhecimento. Tal base de conhecimento é utilizado pelo planejador, como indicado na seta 2, para verificar as condições de ativação e término dos planos e para acessar as crenças sobre as capacidades dos membros do time. As setas 3 e 4, indicam que o planejador pode deliberar sobre a alocação de papéis tanto para o agente acoplado quanto para os demais membros do time. Informações de outros *proxies* podem atualizar o conjuto de crenças, como visto na seta 5, e um agente pode compartilhar informações usando o *TWProxy* como meio, como visto na seta 6.

O planejador do *TWProxy* utiliza o conceito de líder de time para lançar um novo plano. O líder do time pode ser prefixado estaticamente ou definido dinamicamente em tempo de execução. O uso de líder de time evita a necessidade de resolver problemas de conflitos na alocação de papéis, economizando o tempo para reagir às mudanças do ambiente. O líder do time tem as crenças atualizadas e pode lançar de forma consistente um plano de time, porque todos estão comprometidos a compartilhar informações relevantes para o time.

Para alcançar os requerimentos de tempo-real, o processo de alocação de papéis não é inteiramente distribuído. Depois que o líder do time recebe informações sobre as habilidades do time, ele decide sozinho sobre a alocação. Esta é a principal limitação do *TWProxy*, mas também é a principal característica que aumenta a performance na alo-

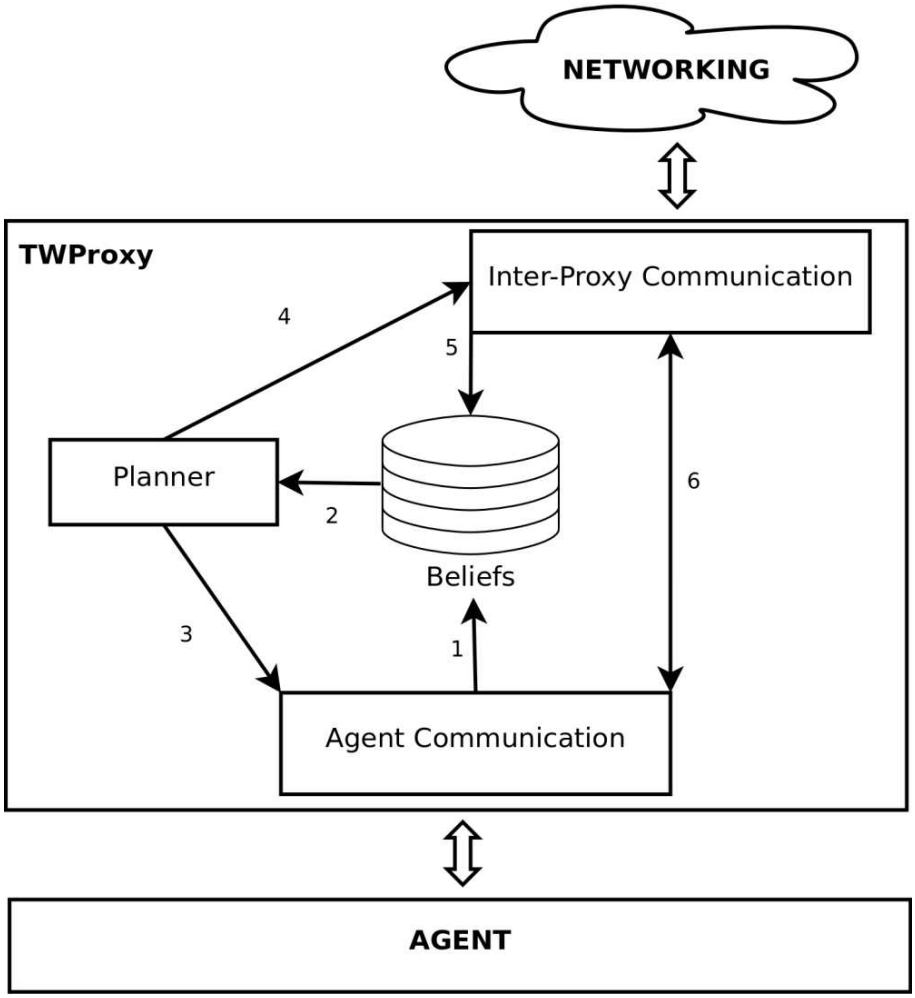

Figura 3.6: *TWProxy organization.*

cação de papéis. São necessárias poucas de trocas mensagens para atualizar o líder do time com as habilidades da equipe, e desta forma o processo é otimizado para alcançar o melhor desempenho.

4 OBJETIVO

Este projeto tem como objetivo geral propor um novo modelo de *teamwork* que atenda aos requisitos impostos por domínios altamente dinâmicos, sacrificando ao mínimo as características gerais dos modelos atuais e possibilitando a formação de times heterogêneos nesses domínios. O *TWProxy* (MONTEIRO; ALVARES, 2008) (MONTEIRO; ALVARES, 2009), como ferramenta, já representa um passo nesse sentido, pois ele foca justamente na manutenção da coerência de time em ambientes altamente dinâmicos. Entretanto, ele possui muitos pontos a serem melhorados, como a autonomia ajustável, afim de se constituir em um modelo genérico de *teamwork* heterogêneo para domínios com requisitos de tempo-real.

Para atingir este objetivo geral, este projeto possui diversos objetivos específicos listados a seguir:

- Identificar as características que levam a maioria dos modelos atuais de *teamwork* a falhar em ambientes bastante dinâmicos.
- Identificar as possíveis restrições a que deverá estar submetido um sistema multiagente situado em um ambiente bastante dinâmico, em virtude de limitações e especializações de algumas de suas características gerais como: organização, comunicação e decisão.
- Avaliar as possíveis formas de interação de humanos com agentes artificiais em um time.
- Balancear as restrições a serem impostas a sistemas multiagentes para a formação de *teamwork* heterogêneo em ambientes altamente dinâmicos, afim de manter o modelo a ser proposto o mais genérico e flexível possível.
- Avaliar as possíveis perdas de qualidade das decisões de time em função do tempo de resposta requerido.
- Propor um modelo geral de *teamwork* para ambientes altamente dinâmicos.
- Implementar um protótipo do modelo proposto.
- Validar o modelo em domínios dinâmicos com requisitos de tempo-real.

5 METODOLOGIA

As etapas previstas para esse projeto são baseadas nas investigações e análises dos modelos atuais, identificando características importantes que devem ser ajustadas para alcançar um modelo genérico de *teamwork* para a classe de domínio proposta. Assim, a revisão bibliográfica apresenta-se como o passo inicial e também persistente ao longo desse projeto. Apesar do foco na sub-área de sistema multiagente conhecida como *teamwork*, outras sub-áreas da inteligência artificial e até mesmo áreas como sistemas de tempo-real e sistemas distribuídos são relevantes para a pesquisa.

A investigação das deficiências encontradas em modelos e ferramentas atuais é um método a ser utilizado para identificar quais características são críticas no tratamento de ambientes altamente dinâmicos. Na modelagem do *E-GAP* (SEEM et al., 2004), por exemplo, existe um termo de custo que penaliza a função a ser otimizada quando ocorre atraso na execução de algum papel. Entretanto, o algoritmo LA-DCOP, apresentado no mesmo trabalho, resolve o *E-GAP* apenas de forma aproximada e não leva em consideração a penalização pelo atraso.

Outra etapa importante é analisar como um sistema multiagente pode ser afetado em função da formação de times em ambientes que possuem requisitos de tempo-real. Avaliar a questão da organização multiagente, identificando se existe ou não necessidade de se fazer alguma restrição. Definir as características de canal de comunicação e os recursos necessários para a comunicação. Também nesta etapa é necessário verificar a forma como as decisões de time precisam ser tomadas, analisando as perdas e ganhos de abordagens centralizadas, parcialmente distribuídas e totalmente distribuídas.

Um grande desafio neste projeto é a identificação das possíveis formas de como humanos podem interagir com agentes artificiais em times, degradando ao mínimo a reatividade da equipe e aumentando a eficiência do grupo. Isso se torna um desafio porque é difícil criar uma compreensão mútua entre membros reais e agentes artificiais. Além do conhecimento compartilhado do time, é necessário que haja de alguma forma uma predição mútua e também que os membros do time se adaptem uns aos outros.

A necessidade de reagir em um tempo de resposta bastante curto pode levar a perda de qualidade nas decisões do time. Por isso, será necessário quantificar o quão está se perdendo em qualidade para atender a um requisito de reatividade. Assim, também é um desafio neste projeto permitir um balanceamento entre a qualidade das decisões e a reatividade do time, lembrando que para ambientes dinâmicos a reatividade influi na qualidade das ações.

A investigação das deficiências dos modelos atuais, a identificação das restrições de organização, a integração humano-agente e o balanceamento entre a qualidade das decisões e reatividade do time, serão executados de duas maneiras principais: pela análise dos modelos teóricos existentes juntamente com os modelos parciais propostos, e pela avaliação de experimentos.

Com base na análise realizada será proposto um modelo geral de *teamwork* para ambientes altamente dinâmicos. Tal modelo pretende possibilitar a definição de vários aspectos de sistemas multiagentes, como suas possíveis organizações, a forma como se efetua a comunição entre os agentes e o comprometimento entre os membros do time. Além de descrever também a forma de interação dos agentes de um time com os membros humanos e como será feito o processo de decisão da equipe.

Para validação do modelo a ser proposto, um protótipo será implementado e aplicado no desenvolvimento de times de agentes para domínios distintos como jogos de computador, resgate simulado em ambiente de desastre e futebol simulado. Serão realizadas análises de desempenho em termo de escalabilidade, eficácia da coordenação e tempo de inconsistência para troca de papéis.

Parte da pesquisa será realizada em uma instituição no exterior. Contatos já foram estabelecidos com o Prof. Dr. Paul Scerri, da Carnegie Mellon University, que mostrou-se bastante interessado no trabalho. Pretende-se também entrar em contato com Prof. Dr. Milind Tambe, chefe do grupo de pesquisa em *teamwork* (TEAMCORE) da University of Southern California.

6 CRONOGRAMA

A Tabela 6.1 apresenta o cronograma de atividades a serem desenvolvidas, conforme citadas abaixo:

1. Revisão bibliográfica.
2. Interação com instituição no exterior para aprofundamento teórico e desenvolvimento parcial da parte experimental.
3. Proposta de tese de doutorado.
4. Desenvolvimento do modelo geral de *teamwork*.
5. Desenvolvimento de protótipo do modelo proposto.
6. Validação do modelo.
7. Escrita de artigos com resultados parciais/finais.
8. Redação da Tese.
9. Defesa.

Tabela 6.1: *Cronograma*

Etapa	2009/1	2009/2	2010/1	2010/2	2011/1	2011/2	2012/1	2012/2
1	xxxxxx	xxxxxx	xxxxxx	xxxxxx	xxxxxx	xxxxxx	xxxxxx	
2			xxxxxx	xxxxxx				
3					xxxxxx			
4				xxxxxx	xxxxxx	xxxxxx		
5					xxxxxx	xxxxxx		
6						xxxxxx	xxxxxx	
7		xxxxxx		xxxxxx		xxxxxx		xxxxxx
8						xxxxxx	xxxxxx	xxxxxx
9								xxxxxx

7 RESULTADOS ESPERADOS

Este projeto tem a expectativa de formar um doutor em computação e de avançar, de forma significativa, o estado da arte em sistemas multiagentes com importantes contribuições na área de *teamwork*. Espera-se que tais contribuições possibilite que problemas envolvendo ambientes altamente dinâmicos sejam resolvidos de maneira mais simples e mais eficiente, possibilitando, inclusive, novas aplicações em tais ambientes.

REFERÊNCIAS

CHALUPSKY, H.; GIL, Y.; KNOBLOCK, C.; LERMAN, K.; OH, J.; PYNADATH, D.; RUSS, T.; TAMBE, M. **Electric elves**: applying agent technology to support human organizations. 2001.

FININ, T.; FRITZSON, R.; MCKAY, D.; MCENTIRE, R. KQML as an Agent Communication Language. In: INTERNATIONAL CONFERENCE ON INFORMATION AND KNOWLEDGE MANAGEMENT (CIKM'94), 3., 1994, Gaithersburg, MD, USA. **Proceedings...** ACM Press, 1994. p.456–463.

GROSZ, B. J.; KRAUS, S. Collaborative Plans for Complex Group Action. **Artificial Intelligence**, [S.l.], v.86, n.2, p.269–357, 1996.

GROSZ, B.; KRAUS, S. **The evolution of SharedPlans**. 1998.

HILL, R.; CHEN, J.; GRATCH, J.; ROSENBLOOM, P.; TAMBE, M. Intelligent agents for the synthetic battlefield: A company of rotary wing aircraft. In: INNOVATIVE APPLICATIONS OF ARTIFICIAL INTELLIGENCE (IAAI-97), 1997. **Anais...** [S.l.: s.n.], 1997.

JENNINGS, N. R. Controlling Cooperative Problem Solving in Industrial Multi-Agent Systems Using Joint Intentions. **Artificial Intelligence**, [S.l.], v.75, n.2, p.195–240, 1995.

JONES, R. M.; LAIRD, J. E.; NIELSEN, P. E.; COULTER, K. J.; KENNY, P. G.; KOSS, F. V. Automated Intelligent Pilots for Combat Flight Simulation. **AI Magazine**, [S.l.], v.20, n.1, p.27–41, 1999.

KAMINKA, G. A. The RoboCup-98 Teamwork Evaluation Session: a preliminary report. In: ROBOCUP, 1999. **Anais...** [S.l.: s.n.], 1999. p.345–356.

LAIRD, J. E.; NEWELL, A.; ROSENBLOOM, P. S. SOAR: an architecture for general intelligence. **Artif. Intell.**, Essex, UK, v.33, n.1, p.1–64, 1987.

LEVESQUE, H. J.; COHEN, P. R.; NUNES, J. H. T. On Acting Together. In: AAAI-90, 1990, Boston, MA. **Proceedings...** [S.l.: s.n.], 1990. p.94–99.

LEWIS, R. L. **Cognitive Theory, SOAR**. 1999.

MARSELLA, S.; ADIBI, J.; AL-ONAIZAN, Y.; KAMINKA, G. A.; MUSLEA, I.; TAMBE, M. On being a teammate: experiences acquired in the design of RoboCop teams. In: THIRD INTERNATIONAL CONFERENCE ON AUTONOMOUS AGENTS (AGENTS'99), 1999, Seattle, WA, USA. **Proceedings...** ACM Press, 1999. p.221–227.

MARSELLA, S.; TAMBE, M.; ADIBI, J.; AL-ONAIZAN, Y.; KAMINKA, G. A.; MUSLEA, I. Experiences Acquired in the Design of RoboCup Teams: a comparison of two fielded teams. **Autonomous Agents and Multi-Agent Systems**, [S.l.], v.4, n.1/2, p.115–129, 2001.

MONTEIRO, I. M.; ALVARES, L. O. A Real-Time Proxy for Flexible Teamwork in Dynamic Environments. In: VII BRAZILIAN SYMPOSIUM ON COMPUTER GAMES AND DIGITAL ENTERTAINMENT - COMPUTING TRACK, 2008. **Anais...** [S.l.: s.n.], 2008. p.83–90.

MONTEIRO, I. M.; ALVARES, L. O. TWProxy: a teamwork proxy for real-time dynamic environments. **VIII Internation Conference on Autonomous Agents and Multiagent Systems (under review)**, [S.l.], 2009.

NAIR, R.; ITO, T.; TAMBE, M.; MARSELLA, S. **RoboCup-Rescue**: a proposal and preliminary experiences. 2000.

PYNADATH, D. V.; TAMBE, M. An Automated Teamwork Infrastructure for Heterogeneous Software Agents and Humans. **Autonomous Agents and Multi-Agent Systems**, Hingham, MA, USA, v.7, n.1-2, p.71–100, 2003.

RICH, C.; SIDNER, C. L. COLLAGEN: when agents collaborate with people. In: FIRST INTERNATIONAL CONFERENCE ON AUTONOMOUS AGENTS (AGENTS'97), 1997, New York. **Proceedings...** ACM Press, 1997. p.284–291.

SCERRI, P.; JOHNSON, L.; PYNADATH, D. V.; ROSENBLOOM, P.; SCHURR, N.; SI, M.; TAMBE, M. **Getting Robots, Agents and People to Cooperate**: an initial report. 2003.

SCERRI, P.; PYNADATH, D.; JOHNSON, L.; SCHURR, R.; SI, M.; TAMBE, M. **A prototype infrastructure for distributed robot-agent-person teams**. 2003.

SCERRI, P.; PYNADATH, D.; SCHURR, N.; FARINELLI, A.; GANDHE, S.; TAMBE, M. Team Oriented Programming and Proxy Agents: the next generation. In: PROGRAMMING MULTIAGENT SYSTEMS, 1., 2004. **Proceedings...** [S.l.: s.n.], 2004.

SCERRI, P.; PYNADATH, D.; TAMBE, M. Adjustable autonomy in real-world multi-agent environments. In: AGENTS '01: PROCEEDINGS OF THE FIFTH INTERNATIONAL CONFERENCE ON AUTONOMOUS AGENTS, 2001, New York, NY, USA. **Anais...** ACM, 2001. p.300–307.

SCERRI, P.; PYNADATH, D.; TAMBE, M. **Towards adjustable autonomy for the real world**. 2003.

SCHURR, N.; MARECKI, J.; TAMBE, M.; SCERRI, P. **Towards Flexible Coordination of Human-Agent Teams**. 2005.

SEEM, P.; FARINELLI, A.; OKAMOTO, S.; TAMBE, M. Token approach for role allocation in extreme teams: analysis and experimental evaluation. **Enabling Technologies: Infrastructure for Collaborative Enterprises, 2004. WET ICE 2004. 13th IEEE International Workshops on**, [S.l.], p.397–402, June 2004.

STEVEN, N. S. **Evolution of a Teamwork Model**. Disponível em citeseer.ist.psu.edu/679191.html. Acesso em: Outubro 2007.

STONE, P.; VELOSO, M. M. Task Decomposition and Dynamic Role Assignment for Real-Time Strategic Teamwork. In: AGENT THEORIES, ARCHITECTURES, AND LANGUAGES, 1998. **Anais...** [S.l.: s.n.], 1998. p.293–308.

SYCARA, K.; PAOLUCCI, M.; VELSEN, M. van; GIAMPAPA, J. **The RETSINA MAS Infrastructure**. [S.l.]: Robotics Institute Technical Report, Carnegie Mellon, 2001. (CMU-RI-TR-01-05).

SYCARA, K.; SUKTHANKAR, G. **Literature Review of Teamwork Models**. Pittsburgh, PA: Robotics Institute, Carnegie Mellon University, 2006. (CMU-RI-TR-06-50).

TAMBE, M. Towards Flexible Teamwork. **Journal of Artificial Intelligence Research**, [S.l.], v.7, p.83–124, 1997.

TAMBE, M. Agent architectures for flexible, practical teamwork. In: NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE (AAAI), 1997. **Anais...** [S.l.: s.n.], 1997.

TAMBE, M.; SHEN, W.; MATARIC, M.; GOLDBERG, D.; MODI, P.; PYNADATH, D.; QIU, Z.; SALEMI, B. **Teamwork in cyberspace**: using teamcore to make agents team-ready.

TAMBE, M.; ZHANG, W. **Towards flexible teamwork in persistent teams**. 1998.

TAMBE, M.; ZHANG, W. Towards Flexible Teamwork in Persistent Teams: extended report. **Autonomous Agents and Multi-Agent Systems**, [S.l.], v.3, n.2, p.159–183, 2000.

VU, T.; GO, J.; KAMINKA, G.; VELOSO, M.; BROWNING, B. MONAD: a flexible architecture for multi-agent control. In: IN PROCEEDINGS OF THE AAMAS¿03, 2003. **Anais...** [S.l.: s.n.], 2003. p.449–456.

YEN, J.; YIN, J.; IOERGER, T. R.; MILLER, M. S.; XU, D.; VOLZ, R. A. CAST: collaborative agents for simulating teamwork. In: IJCAI, 2001. **Anais...** [S.l.: s.n.], 2001. p.1135–1144.