1 Introdução

A tecnologia de software está passando por uma transição de arquiteturas monolíticas, construídas a partir de um projeto único e coeso para arquiteturas compostas por agentes e sistemas multi-agentes semi-autônomos e heterogêneos. Estes agentes são projetados de forma independente e em geral são construídos e gerenciados por organizações distintas e com pouco conhecimento umas das outras [40]. Estas arquiteturas são marcadas pela existência de novas propriedades do nível do sistema (ou inter-agentes) como coordenação [40], adaptabilidade [42], mobilidade [32], segurança [31,60], gerenciabilidade [40] e persistência [42]. Cada uma destas propriedades requer estratégias (ou políticas) que controlem a lógica (isto é seus agentes) e os dados da aplicação.

Dentre os problemas inerentes a tal transição, nenhum é mais sério que a dificuldade de incorporar e compor múltiplas estratégias de controle, o que leva à necessidade de uma abordagem mais sofisticada desde a fase arquitetural de um sistema multi-agente. As funcionalidades básicas de agentes de software já são bastante complicadas, de forma que suas estratégias de controle necessitam ser projetadas e implementadas separadas do seu comportamento básico. Na verdade, os níveis de satisfação dos requisitos de qualidade (por exemplo, capacidade de reutilização e manutenção) de um sistema multi-agente são fortemente dependentes de sua arquitetura [49]. Desta forma, se uma arquitetura que oferece suporte ao tratamento de múltiplas estratégias de controle for escolhida desde o início do desenvolvimento de um sistema multi-agente, torna-se mais simples atingir os atributos de qualidades necessários no restante do desenvolvimento do sistema.

A área de arquitetura de software [49] surgiu na última década como uma área central para engenheiros de software para sistemas complexos. Esta área está preocupada com a definição de estilos e padrões de alto nível para a organização e estruturação fundamental de sistemas de software. Um padrão arquitetural [7] propõe uma solução para um problema recorrente, definindo um conjunto de componentes e subsistemas, assim como regras que organizam os relacionamentos existentes entre eles. Padrões arquiteturais são os blocos de construção de arquiteturas de software de larga escala, que em geral incluem mais de uma instância destes padrões compostos de maneiras arbitrárias [3]. Uma composição específica de padrões arquiteturais, que

ocorre freqüentemente em um dado domínio de aplicação, também pode ser definida como um outro padrão.

No contexto de sistemas multi-agentes, o padrão arquitetural Blackboard [7] tem sido largamente utilizado como uma metáfora útil para o tratamento da comunicação e coordenação de organizações de agentes heterogêneos, provendo baixo acoplamento espacial e temporal [9,28]. A idéia de arquiteturas baseadas em *blackboards* não é nova e foi introduzida no projeto Hearsay II [11]. Atualmente esta mesma idéia tem sido utilizada em projetos comerciais como IBM TSpaces [35] e JavaSpaces.[14]. O padrão arquitetural Blackboard [7] define os componentes e regras deste tipo de arquitetura: múltiplas *fontes de conhecimento* ou *agentes* independentes, cada um implementando uma parte específica da lógica da aplicação, interagem entre si usando o componente denominado *blackboard*; o *blackboard* por sua vez é uma estrutura de dados que é usada como um mecanismo geral de comunicação e coordenação para os diferentes agentes, encapsulando mensagens e informações, e é gerenciado por um componente de *controle*. No entanto, o padrão não especifica de forma explícita como o componente de controle deve lidar com as distintas estratégias de controle necessárias para gerenciar o *blackboard*, nem como separar estas estratégias de controle dos agentes e dados que compõem uma aplicação, o que pode levar a arquiteturas de software multi-agente que são mais difíceis de manter, compreender e reutilizar.

Neste trabalho, é proposto o padrão arquitetural Reflective Blackboard [52,53,54] que é construído a partir da composição de dois outros padrões arquiteturais bem conhecidos [7]: o padrão Blackboard e o padrão Reflection. Na verdade, a combinação do padrão Reflection com outros padrões já foi utilizada com sucesso para definir outros padrões para o tratamento de problemas relevantes [12,21,22]. Como resultado da composição proposta, os componentes do padrão Reflection são utilizados para refinar a estrutura geral definida pelo padrão Blackboard e com o objetivo de promover uma melhor separação de responsabilidades[1]. A separação de responsabilidades é na verdade um dos principais princípios da engenharia de software e é atingida em arquiteturas reflexivas (baseada no padrão Reflection) através da separação do sistema em dois níveis: o nível base e o meta-nível. O padrão arquitetural Reflective Blackboard segue esta organização: enquanto a lógica e os dados da aplicação multi-agente são encapsulados no nível base, o componente de controle é situado no meta-nível. Desta forma, as estratégias de controle do sistema podem ser

[1] Do inglês *separation of concerns*

tratadas completamente separadas de sua lógica e dados. Por sua vez, os agentes da aplicação são projetados de forma independente de suas estratégias de controle. Desta forma, um agente que deseja comprar uma mercadoria em um Marketplace Eletrônico [1,2,27,59] não precisa conter em sua lógica as regras e políticas (estratégias de controle) específicas do Marketplace. Como exemplo destas políticas podem ser citadas, podem ser citadas normas de racionamento que limitam o a quantidade comprada de uma mesma mercadoria por um mesmo agente ou normas que impedem a criação de cartéis formados por agentes vendedores. Esta abordagem é importante no que se refere a aplicações multi-agentes de grande porte uma vez que sua lógica já é bastante complicada devido à complexidade inerente a sistemas multi-agentes de grande porte [23].

O padrão arquitetural Reflective Blackboard é independente de linguagem de programação e pode ser utilizado para minimizar a complexidade causada pela presença de numerosas propriedades inter-agentes (coordenação, mobilidade, persitência e comunicação) em sistemas multi-agentes. O padrão proposto é direcionado em primeiro lugar a engenheiros de aplicações muti-agentes complexas e que precisam definir e implementar as diferentes estratégias de controle que gerenciam seus sistemas. O padrão também pode ser interessante para desenvolvedores de tipos diferentes de infra-estruturas e frameworks baseados em blackboards uma vez que eles poderão incorporar capacidades reflexivas diretamente em seus produtos.

O padrão Reflective Blackboard provê então suporte à implementação de estratégias de controle para os diferentes agentes de um sistema e para as propriedades do nível do sistema. Entretanto, torna-se necessária uma infra-estrutura que dê suporte a estas propriedades e que, ao mesmo tempo, possibilite sua utilização em um sistema baseado no padrão proposto. Desta forma a utilização de estratégias de controle baseadas no meta nível, pode ser utilizada de forma mais simples para coordenar e compor estas propriedades.

Neste contexto, além da definição do padrão arquitetural, também faz parte do objetivo deste trabalho a implementação de uma infra-estrutura para o desenvolvimento de sistemas multi-agentes baseados na arquitetura proposta pelo Reflective Blackboard. Foi então desenvolvido o framework T-Rex (Espaços de **T**uplas **Re**fle**x**ivos) [51,50] que além de ser uma implementação do padrão proposto, oferece suporte a diferentes propriedades do nível do sistema, como mobilidade, comunicação, persistência e coordenação.

T-Rex é baseado em uma variante do padrão Reflective Blackboard denominada Espaços de Tuplas Reflexivos. Esta variante utiliza espaços de tuplas à la Linda [16] como infra-estrutura para o blackboard do nível base. Além disso, eles são também utilizados para armazenar as diferentes estratégias de controle existentes no meta-nível.

As propriedades do nível do sistema em T-Rex (mobilidade, comunicação e persistência) são também implementadas sempre através da infra-estrutura de seus espaços de tuplas. Desta forma, as estratégias de controle podem ser facilmente associadas a estas propriedades, através da reflexão, simplificando sua coordenação e composição. Além disso, a utilização de uma tecnologia única pode ajudar na manutenção e suporte de sistemas multi-agentes que se baseiam na infra-estrutura fornecida por T-Rex.

Não faz parte do objetivo de T-Rex oferecer suporte à modelagem e implementação de funcionalidades internas aos agentes de software. Por outro lado, uma de suas características é poder ser utilizado, independentemente da forma através da qual os agentes são modelados ou implementados.

O objetivo deste trabalho pode então ser resumido em duas atividades distintas e complementares:

(i) Definir um padrão arquitetural para sistemas multi-agentes que dê suporte à implementação de estratégias de controle de maneira separada da lógica básica do sistema;

(ii) Desenvolver uma infra-estrutura para o desenvolvimento de sistemas multi-agentes baseados no padrão proposto e que necessitem de propriedades do nível do sistema como mobilidade, comunicação e persistência.

Além da implementação da infra-estrutura de T-Rex foi desenvolvida também, com o objetivo de validar sua utilização em sistemas com múltiplas propriedades do nível do sistema e múltiplas estratégias de controle, uma aplicação de Marketplace [1,2,27,59]. Nesta aplicação, T-Rex foi utilizado para prover a comunicação entre os agentes, a mobilidade e persistência destes. Além disso, diferentes estratégias de controle foram utilizadas para compor a propriedade de comunicação com a mobilidade e persistência dos agentes. Estratégias de controle também foram utilizadas para implementar atividades de controle do próprio Marketplace.

No restante deste trabalho, após uma apresentação sucinta de alguns conceitos básicos (Capítulo 2), o padrão Reflective Blackboard (Capítulo 1) e T-Rex (Capítulo 1) serão apresentados em detalhes. Ao final, serão apresentadas algumas conclusões e alguns pontos para trabalhos futuros (Capítulo 5).