1

Introdução

Com o passar dos anos, a adoção de sistemas de *software* distribuídos para solucionar problemas complexos tornou-se cada vez mais frequente. Isso se deve, principalmente, ao fato de que há a necessidade de se obter um maior poder de processamento e armazenamento, devido à crescente complexidade dos problemas que esses sistemas são projetados para resolver. De fato, os sistemas distribuídos, através da integração das várias unidades computacionais que os compõem, atendem a essas necessidades. Prova disso é a escolha de uma arquitetura distribuída composta por mais de 15.000 computadores comuns equipados com *software* tolerante a falhas, adotada para ser o pilar do buscador *web* mais utilizado em todo mundo: o Google (Barroso, 2003). Segundo o autor do trabalho, os benefícios são vários, mas o principal deles é atingir um desempenho incrível a uma fração do custo de uma arquitetura semelhante composta por servidores.

Apesar dos benefícios, a adoção de estratégias distribuídas na construção de sistemas de *software* pode gerar diversos problemas. Na maioria dos casos, implica na necessidade desse sistema ser capaz de operar, do ponto de vista lógico, de forma transparente e coordenada nos mais diversos computadores, com características e capacidades possivelmente diferentes.

Considere, então, o desafio de se projetar um sistema distribuído descentralizado. Como cada uma das suas unidades é autônoma no que diz respeito ao processo de tomada de decisões, a desorganização ou descoordenação pode facilmente fazer com que o sistema atinja um estado totalmente caótico, levando-o ao colapso. Portanto, a adoção de soluções distribuídas descentralizadas no projeto de sistemas de *software* implica, obrigatoriamente, na necessidade de se tomar diversas precauções para garantir a organização desses sistemas.

Nesse contexto, arquitetos de *software* têm optado cada vez mais por utilizar estratégias auto-organizáveis para o projeto de sistemas descentralizados. A principal característica de um sistemas auto-organizável é ser dinâmico e adaptativo, capaz de executar sem a interferência do usuário, administrador ou de outros sistemas, adquirindo e mantendo informações sobre o ambiente e o seu entorno sem qualquer controle externo, com o objetivo de ajust-

tar o seu comportamento de modo a garantir o seu correto funcionamento (De Wolf, 2007).

Entretanto, esse processo de auto-organização muitas vezes resulta em comportamentos novos, não previstos, gerados pela interação entre as unidades autônomas do sistema e observáveis apenas no seu nível macroscópico. A esses comportamentos dá-se o nome de **comportamentos emergentes**.

Os sistemas multiagentes, mais especificamente, tornaram-se a alternativa mais difundida na literatura para a construção de sistemas auto-organizáveis (Wooldridge, 2002). Os agentes que formam esses sistemas são unidades autônomas, independentes da existência de outros agentes. Eles possuem um conjunto de capacidades comportamentais que definem a sua competência, um conjunto de objetivos e a autonomia necessária para utilizar suas capacidades a fim de alcançar seus objetivos. Essa tecnologia tem se mostrado cada vez mais flexível e robusta, capaz de se adaptar a ambientes mutantes e manter o seu funcionamento mesmo quando um elemento individual (agente) falha (Bonabeau, 1999).

É interessante observar que, apesar de cada agente possuir certo nível de inteligência, uma inteligência global pode emergir como resultado da interação e comunicação entre agentes que compõe o sistema (De Wolf, 2007). Na maioria das vezes, inclusive, ela é invisível quando se observa o sistema em seu nível microscópico. Pode-se afirmar, então, que a inteligência global de um sistema multiagente constitui um comportamento emergente clássico. Em uma colônia de formigas, por exemplo, as unidades individuais (formigas) desempenham apenas tarefas triviais. Entretanto, considerando a colônia como um todo, percebe-se uma inteligência avançada proveniente da inter-relação entre as formigas por meio dos caminhos de feromônio. E, nesse caso, essa inteligência é o grande responsável pelo sucesso do sistema.

Todavia, é importante ressaltar que nem sempre esse comportamento emergente será benéfico. Quanto maior o sistema (em termos de complexidade, interconexões entre as suas unidades, distribuição geográfica, etc.), maior será a chance de ele apresentar, em algum momento, um comportamento emergente indesejado, prejudicial ao sistema como um todo. Dessa forma, um dos grandes desafios da engenharia de sistemas auto-organizáveis é fazer com que um comportamento dito emergente convirja para um padrão ou tendência desejável que, na maioria dos casos, é um objetivo ou sub-objetivo do sistema.

1.1

Definição do Problema

O constante aumento da complexidade dos sistemas computacionais trouxe consigo uma característica previsível: quanto mais complexo for um sistema, maior será a dificuldade de operá-lo manualmente. Os sistemas auto-organizáveis e, mais especificamente, os sistemas multiagentes, destacam-se como uma solução que busca possibilitar a execução autônoma desse sistema, para que ele dependa cada vez menos da intervenção dos seus administradores e usuários.

Percebe-se, entretanto, que os sistemas multiagentes auto-organizáveis ainda possuem algumas restrições com relação à sua aplicação na indústria; projetar esses sistemas não é uma tarefa trivial, uma vez que não existe uma maneira sistemática de fazê-lo. Atualmente, grande parte das metodologias para a engenharia de sistemas auto-organizáveis é focada no projeto da estrutura microscópica do sistema, isto é, os agentes que o compõe, suas regras de comportamento e interação, etc. (De Wolf, 2007). Essas metodologias são úteis, mas não são capazes de garantir que o sistema projetado se comportará como o esperado, ou seja, não exibirá um comportamento emergente indesejado ou prejudicial.

Faz-se necessária, então, uma forma de garantir as características macroscópicas de um sistema auto-organizável antes mesmo de colocá-lo em funcionamento. No entanto, conseguir isso por meio de sua modelagem microscópica é totalmente inviável. Por outro lado, derivar o comportamento macroscópico do sistema através de métodos analíticos que utilizam os seus atributos microscópicos é praticamente impossível, uma vez que as interações entre as suas unidades autônomas não podem ser descritas formalmente (Wegner, 1997).

Sendo assim, define-se aqui o principal problema que este trabalho busca resolver: a ausência de um método confiável e de simples aplicação que seja capaz de verificar com antecedência se um determinado comportamento emergente desejado (normalmente observável apenas no nível macroscópico) será realmente obtido através da modelagem microscópica proposta para um sistema auto-organizável. Além disso, também é interessante garantir que o sistema não exibirá um comportamento emergente indesejado ou prejudicial. Sem essas duas garantias é muito improvável que se opte por modelar um sistema industrial através de agentes.

O estado da arte demonstra que não há solução simples para esse problema, pois o comportamento que se deseja verificar é muito difícil de ser previsto; isso se deve ao fato de que, sem uma coordenação centralizada, os

agentes cooperam utilizando interações complexas e muitas vezes imprevisíveis, pois cada um deles tem autonomia para decidir quando e como interagir com outros agentes ou com o ambiente (De Wolf, 2007).

Dessa forma, a maior parte das abordagens existentes para a solução desse problema opta por monitorar o sistema durante a sua execução experimental, coletando informações que possivelmente reflitam o seu comportamento emergente com relação às suas características internas (De Wolf, 2007). Entretanto, essas técnicas mostraram-se muitas vezes incompletas ou extremamente complexas, o que dificulta a aplicação em sistemas reais.

1.2 Objetivos

No âmbito da engenharia de *software*, de uma forma geral, os processos de *software* tentam descrever uma sequência de tarefas divididas por etapas, que auxiliem no projeto de sistemas. Essas etapas podem ser executadas consecutivamente uma única vez ou de maneira incremental. Independentemente das particularidades de cada processo de *software*, a verificação (ou teste) do sistema constitui uma etapa importante em todos eles.

A utilização de processos de *software* pré-existentes na engenharia de sistemas multiagentes é restringida por alguns fatores complicadores, pois esses sistemas são muito particulares com relação ao seu comportamento quando em execução. A tentativa de adaptação desses processos esbarra justamente na etapa de verificação; como é possível garantir que o sistema cumprirá realmente o que ele foi projetado para cumprir uma vez que é sabido que comportamentos imprevistos poderão emergir do relacionamento entre os agentes desse sistema?

É importante ressaltar que, no contexto deste trabalho, **verificação** significa o estudo formal ou empírico de um sistema multiagente com a finalidade de averiguar se este sistema é capaz de desempenhar as funções necessárias, de acordo com seus objetivos de projeto. Não há qualquer intenção de se verificar a corretude do código desse sistema, inspecionar sua arquitetura ou analisar seus fluxos de execução.

Dessa forma, este trabalho tem como principal objetivo propor um novo método para a verificação de sistemas multiagentes auto-organizáveis, baseado em simulação experimental (Luke, 2003, Luke, 2004, Luke, 2005) e planejadores em tempo real (Liporace, 2005). São também objetivos secundários deste trabalho:

1. Elaborar um método que seja conceitualmente simples de ser aplicado, sem a necessidade de conhecimentos prévios em formulações complexas,

demonstrações formais ou mesmo programação matemática, para facilitar a sua incorporação a soluções pré-existentes para a engenharia de sistema multiagentes;

2. Apresentar o método proposto na forma de um *framework* de verificação experimental embutido em um ambiente completo e unificado voltado para a engenharia de sistemas multiagentes, onde seja possível:
 - (a) **Projetar** o sistema auto-organizável com o auxílio de um *framework* de sistemas multiagentes que inclua um catálogo de padrões de projeto específicos para este fim (Gatti, 2009);
 - (b) **Observar** o comportamento do sistema através de simulações experimentais com recursos avançados de visualização e monitoração (Luke, 2005);
 - (c) **Verificar** se o comportamento macroscópico apresentado pelo sistema em execução condiz com o que ele foi projetado para fazer;
3. Comprovar a aplicabilidade deste método através da sua utilização em um estudo de caso teórico simplificado: sistema de transporte automatizado de cargas.

1.3

Contribuições

Considerando-se o que foi exposto nas seções anteriores, descrevendo os principais problemas que motivaram este trabalho, bem como os objetivos que se pretendeu atingir durante a sua execução, algumas contribuições podem ser identificadas em suas respectivas áreas de concentração:

1. Engenharia de sistemas multiagentes:

- (a) Desenvolvimento de um método capaz de observar e verificar o comportamento de sistemas multiagentes de forma rápida e eficiente, sem demandar esforços exagerados para a modelagem matemática do sistema que se pretende avaliar como em outros métodos existentes, conforme discutido em (Gatti, 2006);
- (b) Disponibilização de um ambiente completo e unificado voltado para a engenharia de sistemas multiagentes, possibilitando projetar, observar e verificar sistemas auto-organizáveis baseados em agentes.

2. Planejadores:

- (a) Demonstração do conceito de simuladores “caixa-branca” proposto em (Gatti, 2006), isto é, simuladores onde se tem pleno conhecimento das estruturas que os compõem. Isso é um requisito indispensável para o método que este trabalho propõe;
- (b) Aplicação de planejadores em tempo real para a verificação experimental de sistemas multiagentes auto-organizáveis.

3. Computação autônômica: (Kephart, 2003)

- (a) Proposta de um *framework* autônômico para a verificação de sistemas multiagentes auto-organizáveis, isto é, o próprio *framework* que implementa o método apresentado neste trabalho possui características de auto-organização, o que também o enquadra como um sistema autônômico.

1.4

Visão geral

O restante deste trabalho está organizado da seguinte forma:

O capítulo 2 apresenta alguns conceitos teóricos essenciais para a total compreensão do problema apresentado e do processo que se propõe para solucioná-lo. O objetivo, aqui, é fornecer uma noção geral e resumida sobre as duas principais áreas que fundamentam o método de verificação proposto: computação autônômica e sistemas multiagentes.

Já no capítulo 3 são discutidos dois trabalhos relacionados: o primeiro analisa a aplicabilidade de uma abordagem híbrida (em parte formal, em parte empírica) para a verificação de sistemas auto-organizáveis, estabelecendo um breve comparativo conceitual para com as abordagens matemáticas tradicionais; já o segundo trabalho apresentou a possibilidade de se utilizar uma estratégia autônômica baseada no conceito de planejadores para a verificação de sistemas multiagentes. Este último foi o principal fator que motivou e inspirou a concepção e elaboração conceitual do método de verificação proposto neste trabalho.

Em seguida, o capítulo 4 apresenta detalhadamente a arquitetura do método para a verificação experimental de sistemas multiagentes auto-organizáveis proposto neste trabalho, explicando, em linhas gerais, como ele deve ser utilizado. Para facilitar a compreensão do leitor, este capítulo descreve, ainda, a aplicação prática do método proposto no contexto simplificado de um problema bastante conhecido: colônia de formigas (Dorigo, 1996).

Uma segunda aplicação desse método é apresentada no capítulo 5, como estudo de caso. Aqui, o principal objetivo, ao contrário da exemplificação prática utilizada no capítulo anterior, é demonstrar a aplicabilidade da abordagem proposta em problemas um pouco mais complexos e discutir o desempenho do *framework* verificador implementado.

Finalmente, o capítulo 6 analisa os resultados obtidos frente aos objetivos do trabalho e levanta alguns pontos que poderão ser estudados futuramente.