Projeto Arquitetural Automatizado de Sistemas Auto-Organizáveis: Uma Abordagem Baseada em Busca

Lorena do C. Caldas e Sandro S. Andrade
Instituto Federal de Educação, Ciência e Tecnologia da Bahia
Grupo de Pesquisa em Sistemas Distribuídos, Otimização, Redes e Tempo Real
Salvador - Bahia - Brasil
E-mail: {lorecaldas, sandroandrade}@ifba.edu.br

***Resumo*—O aumento da complexidade tem sido citado como um fator limitante para o desenvolvimento e manutenção dos novos sistemas computacionais. Os sistemas que apresentam a capacidade de se auto-organizar têm sido apontados como uma forma de superar tal condição. Este trabalho aborda os temas de Arquitetura de Software e Otimização de Software com o intuito de propor uma solução para automatizar o processo de projeto arquitetural de sistemas auto-organizáveis. Ele apresenta uma nova solução ao problema multiobjetivo de projeto arquitetural para sistemas auto-organizáveis, contendo os objetivos de: maximizar escalabilidade, minimizar sobrecarga de comunicação e minimizar complexidade arquitetural. Para tanto, foram identificados e catalogados treze modelos arquiteturais específicos a este contexto, disponibilizados no espaço de projeto SO:DuSE. Sobre o SO:DuSE foi aplicada a técnica de busca ótima, que combina essas treze arquiteturas através do motor de otimização da ferramenta DuSE-MT. Com isso, três soluções arquiteturais ótimas foram encontradas e validadas para garantir a correta captura das soluções identificadas.**

I. Introdução

Em um passado recente, pesquisadores do campo da Ciência da Computação previram o aumento da complexidade como um fator limitante para o desenvolvimento e manutenção dos novos sistemas computacionais [1], [2]. Características como abertura ao acoplamento de novos componentes (*openness*), escalabilidade e reconfiguração dinâmica passam então a ser necessárias nos novos sistemas computacionais projetados. A possibilidade de trabalhar em ambiente imprevisível torna-se mais um aspecto desejado. Essas características são requeridas porque em um ambiente altamente dinâmico e de estrutura distribuída: *i)* o número de entidades presentes em um sistema pode variar; *ii)* a quantidade de dispositivos operando em um ambiente pode aumentar ou diminuir; *iii)* a topologia de rede precisará ser alterada para atender ao requisito *ii*; e *iv)* o ambiente de operação dos dispositivos pode apresentar falhas ou transformações [3]. Essa situação, também chamada "crise da complexidade" está atualmente se consolidando e os sistemas que apresentam a capacidade de se auto-organizar têm sido apontados como uma forma de superar tais problemas.

Auto-organização é um comportamento que se baseia na interação dinâmica, local e não direta dos componentes de um sistema para reagir aos eventos que aconteçam no ambiente do qual participam. Ele é um mecanismo para gerenciamento de elementos comumente encontrado nos sistemas naturais, mas que tem significativamente crescido, em uso, nos sistemas computacionais.

O mecanismo de auto-organização pode ser encontrado em sistemas categorizados como: sistemas físicos, sistemas vivos e sistemas sociais [4]. Esse mecanismo está presente, por exemplo, no processo de magnetismo físico quando ocorre a atração e a repulsão de matéria; no sistema nervoso humano, que é capaz de manter constantes as condições de funcionamento interno do corpo sem ser controlado conscientemente pelo organismo; no comportamento social dos animais quando se movem em bando para chegar a um determinado local; nas interações sociais humanas quando ocorre a difusão de informações por meio de comunicação indireta; entre outros exemplos.

Projetar sistemas computacionais que apresentem capacidade de auto-organização é uma atividade desafiadora. Eles são sistemas: descentralizados, compostos por unidades autônomas (agentes), que cooperam sem conhecer o mapa do sistema e não sofrem intervenção explícita de um ator ou do ambiente externo. Ainda, são sistemas integram diversas tecnologias, por vezes desconexas entre si e por isso contam com uma ampla gama de requisitos, alta complexidade de relação entre eles, grande amplitude do espaço de solução, presença de *trade-offs* casualmente conhecidos e de fácil uso, além de propriedades constantemente imprevisíveis, dada a dinamicidade deste tipo de sistema. Sendo assim, a atividade de projeto de sistemas auto-organizáveis constitui tarefa dispendiosa ao ciclo de desenvolvimento do *software*. Ela demanda pesquisa e análise das características do sistema e ambiente, revisão das técnicas que podem ser utilizadas para fazê-lo alcançar seus objetivos e definição do modelo do sistema, além de elevado conhecimento e experiência prévia do arquiteto sobre o domínio específico da aplicação, assim como a necessidade de catalogação dos padrões arquiteturais e técnicas voltadas à sua realização. O sucesso do projeto dependem da condução destes diversos fatores que compõem a atividade de projeto de sistemas.

O uso isolado das técnicas convencionais da Engenharia de Software dificulta o projeto de sistemas auto-organizáveis. A combinação entre as diversas técnicas convencionais e outras específicas ao domínio e granularidade de tecnologias envolvidas na composição dos sistemas deste tipo é um meio de atender esta demanda. As técnicas da Engenharia de Software Baseada em Busca (SBSE), matéria da Engenharia de Software que possui foco na aplicação de técnicas de busca combinatorial na resolução de problemas de otimização, também podem auxiliar no projeto dos sistemas auto-organizáveis. Um exemplo de apoio neste sentido é a aplicação de algoritmos evolutivos com o intuito de combinar possíveis soluções de projeto e compará-las a fim de prover ao projeto soluções ótimas e variadas.

Um espaço de projeto é um ambiente abstrato propício ao projeto e definição de soluções arquiteturais. Neste espaço podem ser capturadas diversas soluções arquiteturais com o intuito de compor modelos de sistemas. Um arquiteto de *software* o utiliza para organizar suas ideias de projeto, agrupando as suas características ou combinando-as a fim de criar modelos arquiteturais, tornar suas decisões e comunicá-las ao restante da equipe. A DuSE é uma linguagem de modelagem arquitetural voltada para a construção de espaços de projeto que permitem a captura sistemática dos elementos de projetos arquiteturais relacionados a um domínio de aplicação particular. Este trabalho apresenta um novo espaço de projeto DuSE que oferece suporte ao processo de projeto arquitetural automatizado para sistemas auto-organizáveis. O SO:DuSE realiza a catalogação sistemática de treze das principais características relacionadas a sistemas auto-organizáveis, disponíveis na literatura. Através de seu uso, é possível estender os projetos arquiteturais de sistemas genéricos em sistemas auto-organizáveis.

Os objetivos ao desenvolver este espaço de projeto foram: *i)* catalogar as soluções arquiteturais próprias ao projeto de sistemas auto-organizáveis; *ii)* facilitar a realização da atividade de projeto arquitetural para sistemas deste domínio; e *iii)* propiciar o reuso de suas soluções. Seus resultados foram validados e obteve-se a confirmação da correta captura dos padrões arquiteturais definidos para sistemas auto-organizáveis, pela descoberta de três arquiteturas efetivas ao problema que este trabalho aborda.

O restante deste artigo está organizado como segue: a seção II aborda os conceitos fundamentais utilizados neste trabalho. A Seção III apresenta o novo espaço de projeto proposto neste trabalho. A Seção IV enumera os resultados obtidos. A Seção V apresenta o levantamento de ferramentas semelhantes ao SO:DuSE na literatura atual, possibilitando a consideração da sua relevância em tal cenário. Por fim, a Seção VI reúne as passagens centrais deste trabalho e cita trabalhos futuros.

II. Fundamentos

A auto-organização é um mecanismo descentralizado de coordenação, amplamente presente em sistemas naturais, que permite que múltiplos agentes atinjam objetivos globais através de atuações locais. Ele se relaciona a questões como padrões de comportamento dos seres e equilíbrio de um sistema. Esse mecanismo atua em um ambiente onde podem estar inseridos diversos sistemas. Os sistemas naturais são grupos de plantas, animais ou seres humanos, onde os seus elementos unitários apresentam alguma forma de autonomia, mas relacionam-se para cumprir objetivos comuns. Observando o processo de coordenação auto-organizável do ponto de vista interno a um desses sistemas, a seguinte sequência de eventos fica evidente: *evento 1* – o sistema se instala em um ou mais ambientes; *evento 2* – um dos ambientes sofre alterações que perturbam seus sistemas; *evento 3* – uma parte de um dos sistemas percebe as alterações no ambiente; *evento 4* – esse grupo informa os demais membros do sistema o novo estado do ambiente; *evento 5* – os membros do sistema se reúnem, compartilham informações atuais e prévias do ambiente e tomam decisões a respeito do destino do sistema; *evento 6* – as decisões são implementadas e o sistema se adapta ao ambiente e com isso um ciclo de eventos é concluído. O sistema também pode se adaptar a eventos internos ao sistema (ex.: perda de componentes; chegada de novos agentes; implementação de estratégia ou funções ao sistema, etc.) ou que incidam diretamente sobre ele (ex.: catástrofes, crises em pontos locais, pontos de conflitos com outros sistemas, etc.). O ciclo eventos citado pode também ser conhecido como *feedback loop* ou ciclo de *feedback* e é um método que implementa o controle do mecanismo de auto-organização. Em um sistema auto-organizável, um novo ciclo de eventos pode ser iniciado imediatamente após conclusão de um outro, podendo se repetir a partir do *evento 2*.

Os sistemas computacionais auto-organizáveis são uma categoria de sistemas adaptativos utilizados em ambientes altamente complexos e dinâmicos. Não é uma tarefa trivial tratar destas características, pois são inúmeras e imprevisíveis as variáveis atuantes neste cenário. A forma de coordenação centralizada limita as ações ou até mesmo inviabiliza a responsividade de reação dos agentes do sistema em relação aos aspectos do ambiente. A técnica de descentralização, presente nos sistemas auto-organizáveis, é então uma melhor abordagem para contornar este quesito, pois visa delegar autonomia às partes ou subsistemas do sistema para que, pelo processo de cooperação, os momentos de crise do sistema sejam resolvidos pelos agentes do próprio sistema.

A. Sistemas Auto-Organizáveis

O aumento da complexidade de *software* compeliu a realização de pesquisas direcionadas aos sistemas adaptativos. Essas pesquisas os consideram solução viável à contenção da crise de complexidade. O problema da complexidade de sistemas está relacionado ao aumento exponencial dos esforços para mantê-los conforme a quantidade de componentes, camadas, interações e heterogeneidade de tecnologias que eles apresentam. Ainda, devem ser somados a este problema, aspectos como: uso em larga escala, alta dinamicidade de alocação e alta demanda por disponibilidade de seus serviços. Por conta

disso, os sistemas auto-organizáveis são considerados a opção ideal à contenção desta crise, pois eles contribuem para a atuação autônoma dos agentes por permitir flexibilização. Os novos sistemas também virão a ser não previsíveis, abertos e permitir alterações em sua estrutura em *runtime* [1], [2]. Com isso, sua estrutura deve suportar o comportamento adaptativo – característica fundamental na resolução do problema relatado.

O cenário apresentado introduz os aspectos inerentes aos sistemas computacionais auto-organizáveis. Em sua estrutura os agentes autônomos interagem localmente para tomar decisões que podem provocar mudanças no restante do sistema. Isso acontece sem que ocorra a intervenção direta do ambiente, mas surge das mudanças ocorridas nele que são percebidas pelo próprio sistema [5]. A forma descentralizada de coordenar a adaptação do sistema implica menor controle sobre suas partes, mas garante, com isso, que elas passem a ser entidades capacitadas na resolução de crises pelas quais passa esse sistema. A auto-adaptação, pela cooperação dos agentes, é a característica que torna possível abstrair a complexidade do *software* porque ele ganha a responsabilidade de manter a si próprio.

Um exemplo da aplicação de sistemas computacionais auto-organizáveis é a implementação de um sistema para coordenar um grupo de robôs para que explorem certo ambiente e tracem sua cartografia. Neste caso, o sistema seria projetado para que os robôs utilizassem suas interações locais para repelirem-se mutuamente e desta forma conseguir explorar o maior espaço físico possível deste ambiente. Os sistemas auto-organizáveis também podem ser aplicados em projetos nos campos de: Redes Neurais de Kohonen, SMAs (Sistemas Multiagentes), Computação em Grid, Serviços Emergentes, Comunidades *Web*, *Swarm Robots* (enxame de robôs), Controle de Manufatura, Redes de Sensores, BPIs (*Business Process Infraestructure*) [4] além das Redes Veiculares e outras aplicações.

Os sistemas auto-organizáveis são, ainda, sistemas computacionais de alto desempenho que apresentam propriedades de emergência. Portanto, eles contêm: controle descentralizado, próprio da abordagem arquitetural *top-down*, alta capacidade de adaptação em *runtime*, complexidade resultante, alta escalabilidade, além da implementação de *feedback loop* para regulação interna [6]. Exibem ainda as capacidades *self-**: *self-configuring* (auto-configuração), *self-healing* (auto-recuperação), *self-optimizing* (auto-otimização) e *self-protecting* (auto-proteção), [7].

Os sistemas auto-organizáveis são uma classe particular dos sistemas adaptativos e devem, basicamente: *i)* apresentar uma estrutura contendo múltiplos agentes e *ii)* adaptar-se às mudanças do ambiente sem conhecer seu modelo arquitetural, objetivos ou seu conjunto de políticas explícitas. Os sistemas auto-organizáveis são também: frequentemente não-determinísticos, abertos, funcionam predominantemente de forma não equilibrada e aplicam amplificação auto-catalítica de flutuações[1].

[1]Amplificação auto-catalítica de flutuações: Ação de aumento da auto-modificação, quando do estado de incerteza do sistema, [6].

Os sistemas auto-organizáveis são comumente projetos como sistemas multiagentes. Eles, por vezes, podem ser considerados sistemas distribuídos. A arquitetura dos sistemas multiagentes faz com que as unidades do sistema, os agentes, que funcionam de maneira distribuída, organizem-se de forma apropriada e entre si, para resolver as situações pelas quais passa o sistema. Sobre esta estrutura, cada um dos componentes obtém e guarda as informações locais do ambiente no qual está inserido. A organização dos componentes ocorre então com fundamento nas interações desses elementos e utiliza os dados locais sobre o sistema para alcançar este objetivo. Com isso, as partes não detêm conhecimento do todo do qual participam. No entanto, as partes podem cooperar entre si para concluir uma decisão a ser atribuída a todo o sistema ou porções dele [8]. Por fim, a característica de emergência faz com que eles tenham a capacidade de se adequar às mudanças do ambiente sem que haja a interação direta de um ator sobre ele. Por apresentar todas as características citadas e serem sistemas fortemente baseados em contexto (*awareness*), pode-se considerar que os sistemas auto-organizáveis apresentam grau de ubiquidade.

Pela análise das características citadas, que são comuns aos sistemas auto-organizáveis, pode-se considerar que eles são inerentemente distribuídos, pois é inviável alcançar o nível de auto-organização desejado sem usar múltiplos componentes ligados por uma rede de computadores. O estudo de caso do sistema de transportes do almoxarifado, apresentado na seção IV-A é um exemplo de sistema distribuído. Os sistema auto-organizáveis também apresentam características de ubiquidade, pois são compostos por unidades autônomas, capacitadas a tomarem decisões que mudem o rumo do sistema sem que haja a atuação direta de um ator. O estudo de caso do esquema de evacuação por crise em local fechado, apresentado também na seção IV-A é um exemplo de sistema ubíquo.

B. Engenharia de Software para Sistemas Auto-Organizáveis

O desafio em desenvolver sistemas auto-organizáveis está diretamente relacionado à forma de interação e consequente coordenação dos componentes envolvidos nesses sistemas. Os meios utilizados para controlar a situação baseiam-se em padrões adotados por arquiteturas descentralizadas, o que por vezes reflete padrões encontrados nos sistemas naturais. Por isso o seu processo de projeto não ocorre como nos sistemas convencionais, que podem ser planejados, modelados e implementados com uma única visão. Os sistemas auto-organizáveis utilizam composição de tecnologias para implementar funções nos parâmetros de sua abordagem e por isso é necessário integrá-las corretamente com intuito de promover a coesão.

Sistematizar o conhecimento disseminado na literatura a respeito de sistemas auto-organizáveis é um meio eficiente de diminuir a dificuldade em projetar sistemas deste tipo. Isso pode ser conquistado através da derivação de métodos, processos e tecnologias que sejam genéricos e amplamente aplicáveis à Engenharia de Software para o domínio dos sistemas auto-organizáveis. A abordagem de padrões de projeto facilita essa sistematização. Aplicar padrões de projeto é um

método de apoio ao projeto de sistemas considerado confiável, pela Engenharia de Software, porque eles constituem soluções amplamente difundidas e por isso validadas.

Atualmente a Engenharia de Software para sistemas auto-organizáveis oferece métodos que tornam possível a realização de projeto arquitetural para a construção de sistemas deste tipo. Arquiteturas de referência, catálogos de padrões arquiteturais e catálogos de estilos arquiteturais são exemplos de meios que permitem a exploração de características próprias a esses sistemas e a composição de seus modelos arquiteturais [9].

As abordagens de projeto arquitetural atuais apresentam deficiências que limitam o alcance de melhor desempenho na execução da tarefa de modelagem arquitetural. Algumas delas são: informação não-estruturada ou não específica ao domínio de auto-organização, informação difícil de ser consumida por máquinas, dificuldade em possibilitar a evidência de *trade-offs*, susceptibilidade ao viés e visão parcial do espaço de problema e solução.

O objetivo da Engenharia de Software para sistemas auto-organizáveis é realizar a prospecção de tecnologias para que seja realizada a correta sistematização do conhecimento de projeto, implementação, teste e evolução de sistemas auto-organizáveis. A Engenharia de Software para sistemas auto-organizáveis tem também a missão de tornar possível desenvolver sistemas deste tipo de maneira repetível e efetiva [10]. O conhecimento a respeito deste assunto está difundido na literatura e isso torna mais difícil encontrar as informações mais relevantes à construção de sistemas auto-organizáveis. Esse fator contribui significativamente para que a tarefa de projeto arquitetural de sistemas auto-organizáveis seja complexa de ser realizada. Qualquer forma de centralizar essas informações mais relevantes e disponibilizá-las de maneira sistematizada tornaria mais fácil executar a tarefa de projetar sistemas deste tipo.

1) Aspectos da Engenharia de Software para Sistemas Auto-organizáveis: A Engenharia de Software divide a abstração da construção de sistemas auto-organizáveis em três aspectos. Essa decomposição visa estabelecer um processo de normatização minimamente viável ao desenvolvimento dos sistemas auto-organizáveis. Neste processo, cada aspecto corresponde também a uma etapa do ciclo de vida do desenvolvimento desses sistemas. A seguir estão descritos os aspectos citados [10].

a) Aspecto Teórico: este aspecto visa conduzir a reflexão a respeitos dos fatores que permeiam questões do tipo: "Como abstrair e representar o sistema de maneira acurada?"e "Quais os mecanismos podem ser identificados e ser utilizados na implementação dos sistemas auto-organizáveis?". A resolução destas questões tende a direcionar o correto planejamento e elaboração dos sistemas auto-organizáveis. Este aspecto também subdivide-se em 2 assuntos, que são apresentados logo abaixo.

- Abstração e Modelagem: Existem três métodos para abstrair a formação de estrutura dos sistemas auto-organizáveis. São elas:

  – Equações Diferenciais: Este é um conceito matemático utilizado para modelar comportamentos dinâmicos. Este método permite a verificação dos mecanismos de interação entre os agentes do sistema para tornar possível isolá-los e tratá-los. O seu uso está voltado à modelagem da característica de emergência do sistema, mais especificamente à sincronização da interação entre os agentes. Para tanto uma topologia é definida para a rede e cada unidade do sistema ganha uma posição estratégica.
  Os agentes são divididos em grupos, onde os pares de correlação são fixos. Isso faz com que seja possível realizar formações padronizadas entre elas e outros fenômenos simples de ação coletiva. No entanto, esta condição também limita a diversificação dos tipos de componentes habilitados para uso no sistema, já que os componentes precisam ter níveis de conectividade compatíveis para comunicarem-se.

  – Celular Automata: Método de simulação adequado aos sistemas físicos. Este modelo assume a seguinte visão: o mundo é uma grade discreta que contém *sites*; cada site da grid pode estar em uma das possibilidades do grupo de estados discretos. A forma de interação entre os elementos do modelos é: cada *site* comunica-se localmente com seus vizinhos e por isso ele não interfere diretamente na situação dos agentes fora deste contexto; todos os sinais de comunicação iniciados precisam ser propagados a todos os vizinhos conectados. Para tanto, ele requisita a atualização da sincronização entre os seus elementos.

  – Modelo Baseado em Agentes: É uma metáfora natural que precisa ser aplicada se: *i)* os componentes do sistema ou suas interfaces variam muito umas das outras (heterogeneidade); e *ii)* apresentam uma variedade muito extensa de possíveis comportamentos e estratégias, como por exemplo, a tomada de decisão (dinamicidade). O modelo baseado em agentes não requere a adoção de uma topologia física para a estruturação de um sistema e por isso ele flexibiliza a forma de interação entre seus agentes, quando comparado ao modelo celular automata. Pesquisas a respeito deste campo comumente definem os sistemas multiagentes (SMAs) como forma de abstração para modelar sistemas auto-organizáveis. No entanto, o nível de composição/complexidade deste modelo pode dificultar o acesso ao núcleo do(s) fenômeno(s) tratado(s) e com isso pode interferir diretamente nas especificações e grau de acurácia do modelo do sistema. Este trabalho fundamenta o funcionamento dos sistemas auto-organizáveis sobre a estrutura do modelo baseado em agentes, conforme descrito na seção III.

- Mecanismo de *Self-Organization*:
  O mecanismo de auto-organização é responsável por guiar o comportamento das entidades empregadas no sistema em questão. Isso envolve condições de como agir, interagir e restringir os resultados a respeito dos resultados dos eventos de emergência do sistema. Existem duas linhas de inspiração ao mecanismo de auto-organização: sistemas biológicos e sociedades humanas, já que ambas apresentam aspectos próprios à auto-organização dos sistemas baseados em contexto. Ele pode também apresentar tipos especiais, os quais contém princípios específicos a serem tratados pela Engenharia de Software. Este aspecto está intimamente relacionado à proposta inserida neste trabalho, conforme descrito no decorrer do artigo.

b) Aspecto Tecnológico: o objetivo atual dos profissionais de Engenharia de Software que planejam e projetam os sistemas auto-organizáveis atuais está em retirar um pouco do foco sobre a visão micro do sistema para visualizar o comportamento macro dele. Essa é uma atividade difícil de ser realizada, por vezes pode se tornar inviável realizá-la dado o cenário imprevisível no qual estão inseridos os sistemas deste tipo. Na visão micro a estrutura e comportamento dos agentes do sistema fica em evidência enquanto a visão macro analisa os efeitos dos agentes sobre os subsistemas ou o sistema inteiro. Uma boa estratégia para conseguir dar vazão à visão macro no projeto de sistemas auto-organizáveis é dividir as tarefas citadas em dois grupos de pensamento, os quais são apresentados a seguir.

- Metodologia:
  A metodologia é um aspecto de Engenharia que define a forma com que um *software* será construído. Para escolher a metodologia adequada à construção do sistema é necessário responder às seguintes perguntas: *a)* Essa metodologia fornece conhecimento sobre como construir o sistema? *b)* Quais os princípios e processos necessários ao seu desenvolvimento? Respondê-las é uma forma de avaliar a real eficácia da metodologia, considerando sua relação com o projeto de sistemas.

  Atualmente os arquitetos de *software* têm enfrentado uma grande dificuldade em representar o relacionamento entre os aspectos micro e macro desses sistemas (aspectos de emergência e coordenação dos agentes), assim como representar o comportamento dos agentes, já que eles são autônomos. A resolução desta questão implica em aumento do desempenho dos sistemas auto-organizáveis. Este é um exemplo de cenário onde a aplicação de metodologia propícia pode levar a bons resultados.

- Implementação:
  Os aspectos de autonomia dos agentes e emergência do sistema trazem desafios à implementação dos sistemas auto-organizáveis. O desenvolvimento dos sistemas deste tipo deve ter atenção de três visões distintas:
  - *Middleware* = O *Middleware* deve ser utilizado para dar suporte à implementação dos agentes, pois eles apresentam as características de auto-modificação e interagem com pares não fixos. O *middleware* pode também ser utilizado como o meio de propagação das informações depositadas no sistema entre os agentes e ser assim um meio de interação entre eles. No modelo computacional bio-inspirado, apresentado na seção III, os elementos infraestrutura e os agentes de infraestrutura cumprem exatamente essa função estipulada para o *middleware*.

  - Arquitetura = O projeto de arquitetura de sistemas é responsável por disponibilizar um *framework* capaz de guiar a implementação do sistema. Uma questão em aberto no projeto de arquitetura de sistemas auto-organizáveis é preparar uma abordagem construtiva que permita a utilização posterior. Essa abordagem deve observar as funcionalidades dos componentes do sistema e suas formas de coordenação a nível micro e macro. O SO:DuSE especifica padrões que implementam funções próprias aos sistemas auto-organizáveis, considerando as visões de agente e de sistema (abordagem agente-ambiente).

  - Padrões de Projeto = O emprego dos padrões de projeto sobre uma abordagem de projeto arquitetural propicia o reuso de soluções sistematizadas e esse é justamente o foco de atuação do SO:DuSE. Até o momento, pesquisadores do campo de Search-Based Software Design (SBSD) ou Projeto Arquitetural Baseado em Busca, exercem esforços em disponibilizar um conjunto de padrões de projeto genéricos que facilitem o reuso dos mecanismos de auto-organização nos sistemas de domínio específico. Atualmente estão disponíveis somente catálogos que especificam os padrões arquiteturais em linguagem natural.

- Testes:
  Os sistemas auto-organizáveis ainda precisam de um processo de garantia da qualidade adequado que lhes ofereça suporte. Faz-se necessário criar um método eficaz e prático capaz de antecipar e verificar o comportamento emergente e impactos. A sua característica de imprevisibilidade e a autonomia dos seus componentes são outros fatores que dificultam a realização de testes adequados aos sistemas auto-organizáveis.

c) Uso de Ferramentas: as ferramentas disponíveis ao apoio e desenvolvimento dos sistemas auto-organizáveis podem ser divididas em três categorias: *i)* ferramentas de desenvolvimento de sistemas auto-organizáveis; *ii)* plataformas de simulação de sistemas auto-organizáveis; e *iii)* ambientes de operação dos sistemas auto-organizáveis.

As ferramentas de suporte ao desenvolvimento de sistemas

auto-organizáveis contêm um banco de dados que guarda os possíveis estados dos agentes. Esses estados são refinados pelos mecanismos, quando eles entram em ação. Os estados são escolhidos pela política em uso no momento e são executadas pelos agentes quando o sistema está em operação. Essas duas técnicas estão presentes dentre as treze arquiteturas disponibilizadas neste trabalho.

As ferramentas de simulação de sistemas auto-organizáveis disponibilizam um ambiente onde os mecanismos de auto-organização podem ser modelados e simular sua real execução. Desta forma é possível observar os resultados obtidos para aquele sistema ou parte dele. Já as ferramentas de implementação de sistemas auto-organizáveis proporcionam um ambiente distribuído que permite exercitar esse tipo de sistema. Elas provêm funções de implementação a nível micro, como: comunicação, segurança, gerenciamento de recursos e mecanismos de escalonamento de serviços.

2) Técnicas de Projeto Arquitetural para Sistemas Auto-organizáveis: O alto grau de composição de tecnologias dos sistemas auto-organizáveis faz com seja necessário observar e administrar a interação simultânea entre os vários mecanismos que deles participam, incluindo no momento da modelagem arquitetural para sistemas deste tipo. Isso se deve ao fato de que as técnicas convencionais da Engenharia de Software costumam tratar da aplicação de apenas um mecanismo por vez (uma função ou ação incidente sobre um mecanismo por vez), em dado contexto. Combinar agentes desconexos em seus tipos e na combinação entre suas funções, por vezes também desconexas, não é algo facilmente tratável pela aplicação dessas técnicas. É necessário utilizar técnicas específicas ou a combinação entre essas duas vertentes para oferecer o suporte adequado ao modelo de desenvolvimento de sistemas auto-organizáveis.

Ao se trabalhar com a "composição de sistemas complexos de larga escala" – caso dos sistemas auto-organizáveis – passa a ser mandatório adotar meios específicos à resolução de alguns desafios envolvidos nos processos que envolvam seu mecanismo de auto-organização. Esses processo devem atender à coordenação de múltiplos eventos, seus efeitos locais e globais, observando a integração entre as funções dos agentes e os diversos eventos que ocorrem de forma simultânea sobre o ambiente. A importância dada à integração entre as funções dos agentes é o ponto principal que diferencia a aplicação das técnicas de Engenharia de Software para Sistemas Auto-Organizáveis em relação àquelas técnicas de Engenharia de Software convencionais. É deste ponto que partem as funções globais do sistema, provenientes da combinação entre as diversas funções dos agentes. Existem basicamente três formas de tratar a modelagem deste aspecto. São elas: reduzir o fenômeno as primitivas, combinar meios da composição com resultados previstos e tornar os detalhes da complexidade do sistema transparentes aos subsistemas macro.

A redução do fenômeno as primitivas deve envolver aquelas primitivas que contenham propriedades e interfaces bem definidas. Este é um esforço voltado à reutilização e combinação dos fragmentos do fenômeno, já que esta é uma abordagem que visa a decomposição do método para este fim. Definir os vários mecanismos da auto-organização como padrões arquiteturais é um meio de alcançar a decomposição do método em fragmentos. A criação do SO:DuSE tem foco justamente na aplicação desta técnica para prover a facilitação, reuso e combinação das características dos sistemas auto-organizáveis como meio de os projetar.

A correta escolha dos *frameworks* a utilizar e a definição do projeto de arquitetura é uma forma de resolver o aspecto da combinação entre os meios de composição com os resultados previstos. No entanto, isso não implica resolver todos os problemas dos efeitos oriundos das combinações das funções dos agentes, pois é praticamente inviável prever todas as possibilidades. Para resolver este entrave é necessário realizar avanços na técnica de análise formal e voltá-la especificamente aos métodos da auto-organização. O aspecto de transparência dos detalhes da complexidade do sistema é outro fator considerado fundamental na modelagem de sistemas artificiais e que necessita de avanços da análise formal para que seja tratado em sua completude.

O método de análise formal voltado especificamente ao mecanismo da auto-organização está diretamente relacionado a uma ou mais das três dimensões de projeto. Essa dimensões são consideradas críticas, sendo elas: *i)* emergência, a dimensão vertical que coordena a comportamento de baixo nível e de alto nível do sistema; *ii)* organização, a dimensão horizontal responsável por relacionar as entidades de igual nível umas com as outras; e *iii)* dinamicidade, dimensão temporal que explora a forma como o sistema se desenvolve no decorrer de seu período de operação [5].

Todos estes métodos contribuem para a diminuição da complexidade em projetar sistemas auto-organizáveis. Mas delas apresentarem algumas limitações, é necessário ao arquiteto dominá-las, assim como ao negócio do sistema, para aplicá-las. Dito isto, a centralização de informações a respeito de características relevantes à construção de sistemas auto-organizáveis, a sistematização e disponibilização delas representam importantes esforços para efetivar avanços neste sentido e com isso permitem impulsionar o correto desenvolvimento de sistemas deste tipo.

C. Engenharia de Software Baseada em Busca

Inúmeros problemas atuais da Engenharia de Software podem ser reduzidos àqueles encontrados no campo da Otimização de *Software*. Para tanto, faz-se necessário modelá-los como um espaço de busca e aplicar funções objetivo sobre eles. Um espaço de busca é um plano ou um espaço multidimensional matemático próprio à combinação de diversas características de um problema. Dessas combinações resultam as suas soluções, mas elas são selecionadas através do uso de uma ou mais funções-objetivo. Uma função-objetivo é uma função matemática responsável pela avaliação das soluções encontradas sobre um espaço de busca. A(s) função(ões) objetivo é(são) aplicadas diversas vezes sobre o espaço de

busca até que o processo de seleção de soluções ótimas seja finalizado. Com a modelagem do problema sobre um espaço de busca e aplicação das funções-objetivo, os problemas da Engenharia de Software de âmbito mais genérico passam a configurarem-se como problemas específicos da SBSE.

Se considerarmos que os problemas da Engenharia de Software apresentam recursos e objetivos envolvidos, então basta encontrar um meio de formatá-los com uma representação matemática para assim atribuir-lhes alguma heurística disponível e desta forma obter soluções ótimas aos problemas. É o que acontece, por exemplo, quando se tem grupos de requisitos definidos para construir um sistema e deseja-se, com o desenvolvimento desta aplicação, satisfazer tanto ao cliente quanto trazer lucros à empresa. O problema, então, estaria relacionado à seguinte pergunta: "Qual é o conjunto de requisitos que equilibra o custo de desenvolvimento de *software* e satisfação do cliente?". Os recursos desse problema poderiam ser: os conjuntos de requisitos, o custo do desenvolvimento de cada um deles e o grau de satisfação do cliente. Já as suas funções-objetivo poderiam ser duas: *i)* diminuição dos custos e *ii)* aumento da satisfação do cliente. E desta forma, um problema de domínio genérico da Engenharia de Software passaria a estar configurado como um problema da SBSE [11].

Os problemas de busca relacionados ao campo da SBSE podem corresponder aos diversos aspectos da Engenharia de Software e atender aos temas de: requisitos, projetos, testes, gerenciamento e refatoração. Cada qual relaciona-se à SBSE com seu domínio específico e acaba por definir um ramo próprio para sua atuação. O ramo da SBST (*Search Based Software Testing*), por exemplo, é um destes casos, pois seus problemas de busca têm aplicação singular no tema de teste de *software*. Um exemplo de problema da SBST é a seleção de casos de teste para a realização de testes de regressão. O problema se enquadra na minimização de recursos, os cenários de testes, e na maximização de cobertura, os módulos do *software*. Já a SBSD trata daqueles problemas da SBSE de domínio do projeto arquitetural de *software* (*architecture design*). Um problema relacionado à SBSD poderia ser: estruturar a arquitetura do sistema de forma a melhorar seu aspecto de manutenção [11].

Este trabalho aborda o tema de projeto arquitetural, tendo a aplicação do seu problema no campo da SBSD. Ele parte do pressuposto de que o centro de todo o sistema computacional é sua arquitetura [12]. Através de seu projeto, os componentes do sistema são dispostos de tal forma que possam realizar suas funções específicas e comunicarem-se com os demais componentes e aspectos do próprio sistema para assim disponibilizar serviços. Um arquiteto de *software* aplica seu conhecimento e experiência no trabalho de projeto arquitetural de um sistema. Para tanto, ele deve considerar o domínio, os requisitos, funcionais e não-funcionais, o nível de coesão ou integração entre seus elementos, o nível de qualidade a ser alcançada pelo sistema, a abordagem a ser seguida, além dos métodos e ferramentas disponíveis para uso na modelagem da arquitetura.

Com todos esses aspectos envolvidos, há a demanda pela automatização da tarefa de projeto arquitetural. Ao condicionar essa demanda aos sistemas auto-organizáveis, o arquiteto ou engenheiro de *software* ganha a possibilidade de direcionar esforços aos aspectos mais relevantes e relacionadas ao comportamento de alto nível do sistema, demandando à ferramenta àqueles de baixo nível. No entanto, esta não é uma tarefa fácil de ser conquistada, já que a ferramenta deve considerar todos os detalhes citados anteriormente para realizar a construção do projeto arquitetural. Ela deve também estar programada para reconhecer a semântica intrínseca ao modelo, ter acesso a uma variedade de alternativas de projeto e estar habilitada a balancear todos os fatores que intervenham na qualidade do sistema [12].

D. O Método DuSE

Este trabalho está embasado no processo de projeto arquitetural automatizado apresentado em [13]. A proposta deste trabalho é apresentar um meio de tornar esse processo específico ao domínio dos sistemas auto-organizáveis.

A DuSE é uma linguagem de modelagem arquitetural própria à escrita de espaços de busca. Ela é também um metamodelo que define a forma de escrita de modelos arquiteturais na linguagem DuSE. O seu processo está dividido em 4 partes. Existe uma parte direcionada à especialização do processo a um domínio. O SO:DuSE foi derivado do metamodelo DuSE e se integra à parte citada, pois ela se recebe e integra os modelos específicos suportados pelo DuSE-MT [9], [14].

Pode-se dizer que um espaço de projeto é um ambiente propício ao desenvolvimento de projetos arquiteturais que envolve características de um projeto como: dimensões de projeto e seus pontos de variação. Considerando-o como um problema matemático, ele toma a forma de um plano ou um outro espaço contendo mais vertentes, a depender da quantidade de fatores a serem combinados. Ele deve conter como eixos as características de um sistema (*design dimensions*) e cada qual pode conter vértices, que simbolizam decisões específicas ao domínio de um problema (*variation points*). Geralmente um espaço de busca é escrito para representar um problema matemático, seja ele próprio à alocação de recursos, à otimização de lucros ou qualquer outro objetivo que perpasse a SBSE. No caso deste trabalho, um espaço de projeto foi escrito para contemplar as soluções que passarão a compor novos projetos arquiteturais para sistemas auto-organizáveis.

O DuSE-MT é uma ferramenta que suporta o processo automatizado de projeto arquitetural para modelos arquiteturais DuSE. Os modelos devem ser de um dos seguintes tipos: UML ou MOF. Tais modelos são refatorados com base nas características estruturadas sistematicamente no espaço de projeto para este fim. Para tanto, esse espaço de projeto precisa ser escrito na linguagem de modelagem arquitetural DuSE.

O DuSE-MT oferece suporte ao projeto de modelos arquiteturais de qualquer domínio de sistema. Atualmente ele conta com a possibilidade de integração com um espaço de projeto próprio aos sistemas *self-adaptive*, o SA:DuSE. Já o SO:DuSE foi definido para expandir esta oferta aos sistemas auto-organizáveis. O espaço de projeto SO:DuSE parametriza

os padrões arquiteturais básicos e alguns padrões arquiteturais compostos em médio nível, definidos no trabalho [3] apud [15] e [16] para atender aos principais aspectos de auto-organização.

O algoritmo ótimo NSGA-II é atualmente utilizado no motor de otimização do DuSE-MT. No entanto, a ferramenta pode ser modificada para aplicar qualquer outra abordagem de interesse. A arquitetura do DuSE-MT é estruturada por *plug-ins* e por isso basta alterar uma parte dela para aplicar uma nova abordagem ao seu motor de otimização. O algoritmo NSGA-II foi escolhido por ser um algoritmo ótimo (meta-heurística) adequado ao problema que é atacado pelo SO:DuSE. Ele é um algoritmo genético: um algoritmo que aplica sucessivas combinações e seleções sobre uma população de soluções para escolher aquelas mais propícias e variadas, conforme contexto do problema. Os algoritmos genéticos foram criados com inspiração na teoria da evolução pela escolha seletiva proposta pelo cientista Charles Darwin. O algoritmo NSGA-II representa importante contribuição aos trabalhos SBSE já consolidados [17].

Este trabalho apresenta uma nova solução ao problema multiobjetivo de projeto arquitetural para sistemas auto-organizáveis. A solução é o SO:DuSE, um novo espaço de projeto especificado na linguagem DuSE propício à configuração arquitetural de modelos de sistemas genéricos em modelos de sistemas auto-organizáveis. Através de seu uso, sobre a ferramenta DuSE-MT, é possível remodelar um projeto arquitetural que não apresente coordenação de elementos para um segundo tipo que apresente coordenação descentralizada de elementos. Isso é possível porque o motor de otimização, genérico, do DuSE-MT identifica as características específicas aos sistemas auto-organizáveis descritas no SO:DuSE e modela a arquitetura original do sistema para esta nova formatação.

O SO:DuSE captura em suas dimensões de projeto as características mais proeminentes ao assunto de sistemas auto-organizáveis especificadas na literatura afim. Ele relaciona aspectos como infraestrutura de comunicação, técnicas de comunicação, coordenação de movimentos e coordenação de tomada de decisão.

Através de seu uso, o SO:DuSE viabiliza o projeto arquitetural automatizado para sistemas auto-organizáveis e facilita a identificação dos *trade-offs* envolvidos no projeto de arquiteturas deste tipo. Ao final da refatoração propiciada pelo SO:DuSE, a ferramenta DuSE-MT apresenta ao cliente todas as possibilidades de modelo arquitetural próprios a sistemas auto-organizáveis. Essas soluções são encontradas tendo como base o projeto arquitetural original, que é modificado múltiplas vezes até que sejam encontradas soluções arquiteturais viáveis. Desta forma, é possível ao arquiteto observar todas as opções de soluções arquiteturais encontradas pela ferramenta e selecionar aquela que for mais adequada ao seu objetivo pretendido.

A figura 1 apresenta o fluxo detalhado das operações do processo DuSE. O fluxo é dividido em duas vertentes: *a)* concepção do espaço de projeto e *b)* uso do espaço de projeto. O processo é iniciado na vertente **a)** quando ocorre a tarefa de "Especificação de profile UML de domínio específico" e é finalizada quando é encontrada uma "Arquitetura final a ser adotada", após a realização da atividade de "Seleção de arquitetura final a partir do Pareto-front", na vertente **b)**. As 3 atividades da vertente de **concepção do espaço do projeto** devem ser guiadas pelo *conhecimento de projeto do domínio específico*, para que ocorra a correta captura das informações necessárias à refatoração do modelo arquitetural inicial e a otimização das soluções encontradas. Elas foram realizadas nas etapas de projeto e implementação deste trabalho, nos quais foram, respectivamente, criados os modelos UML referentes às arquiteturas iniciais e arquiteturas sobrescritas e o arquivo XMI contendo as dimensões e pontos de variação arquiteturais estruturados conforme linguagem DuSE. Essas informações podem ser encontradas na seção III.

As atividades da vertente **b)** seguem o seguinte fluxo de operação: criação do modelo anotado representando o sistema inicial, com base nas anotações de domínio específico extraído do profile UML realizado na vertente **a)**. O modelo do sistema inicial é utilizado como recurso de entrada na atividade de *Validação das instâncias das dimensões do projeto* nos moldes do Espaço de projeto. Este, por sua vez, é o recurso de entrada na atividade de *Otimização da arquitetura inicial*, com vistas nos Parâmetros de otimização. Por fim, as Arquiteturas Pareto-ótimas são utilizadas na entrada da atividade de *Seleção arquitetural final a partir do Pareto-front* para conduzir a seleção da Arquitetura final a ser adotada. Se necessário, é possível repetir toda a operação desta vertente até que o objetivo seja alcançado. As atividades da vertente de **uso do espaço de projeto** foram realizadas na atividade de validação deste trabalho. Essas informações podem ser encontradas na seção IV.

As subseções a seguir irão apresentar a nova abordagem proposta neste trabalho a níveis mais detalhados. Serão apresentadas informações detalhadas a respeito da implementação do projeto em questão.

E. O Espaço de Projeto SO:DuSE

O SO:DuSE disponibiliza um espaço de projeto voltado ao domínio dos sistemas auto-organizáveis. Ele reúne os principais padrões arquiteturais descritos na literatura a respeito do tema auto-organização arquitetural para *software*. A utilização de padrões arquiteturais permite o reuso desta solução e por isso esta abordagem foi escolhida. Eles também são resultado de práticas já testadas, o que tornam as suas implementações confiáveis.

Os tópicos a seguir descrevem algumas das características relacionadas à estrutura do espaço de projeto SO:DuSE. O tópico II-E1 enumera as dimensões do espaço de projeto SO:DuSE. O tópico III-2 relaciona os aspectos quantitativos associados às dimensões e pontos de variação do espaço de projeto.

1) Dimensões de Projeto: O trabalho [18] apresenta 206 padrões arquiteturais redundantes para sistemas SMAs capturados entre os anos de 1998 e 2012. Eles são dispostos em grupos: Inspiração, Abstração, Foco e Granularidade. O

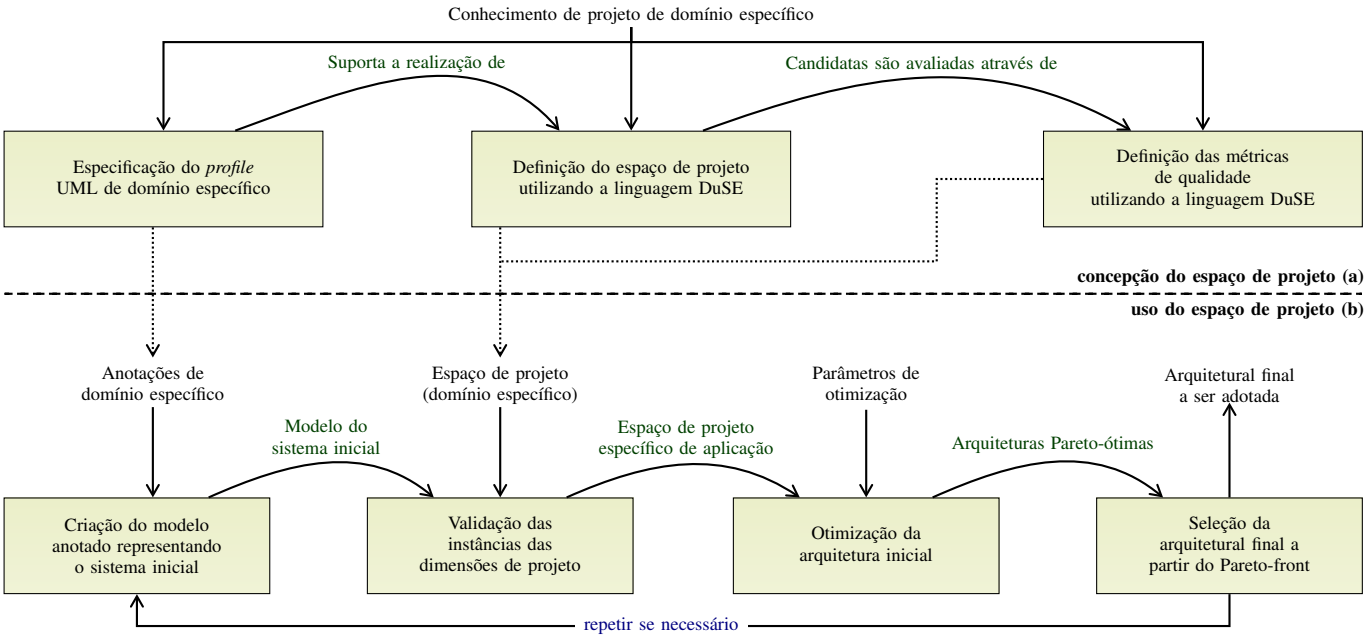

Figura 1. Fluxo de operações do processo DuSE

grupo Inspiração é dividido em 4 categorias: *i) inspiração natural*, que se relacionam aos padrões encontrados na natureza, com o exemplo do padrão *Gradient*, utilizado por colônias de formigas; *ii) inspiração social*, que se relaciona com a característica de emergência encontrada na interações humanas ou animais realizadas no convívio em grupo; *iii) inspiração humana* que se relaciona às interações entre as pessoas, tendo como exemplo o padrão *Gossip* (fofoca); e por último, *inspiração de artefatos*: que se relacionam aos objetos que as pessoas utilizam para dar suporte às suas tarefas, a exemplo da abordagem *Blackboard* (quadro negro).

O grupo Abstração classifica os padrões arquiteturais como conceituais ou concretos. Ambos podem ser refinados em subcategorias relacionadas às fases do ciclo de vida de desenvolvimento do *software* em que podem ser utilizados. Os padrões conceituais são voltados ao uso nas fases iniciais do ciclo. Já os padrões concretos, a exemplo do padrão *Digital Pheromone*, devem ser utilizados nas fases de projeto e implementação do *software*.

O grupo Foco é divido em 2 categorias: estrutural e comportamental. Os padrões arquiteturais elencados na categoria Estrutural lidam com o funcionamento dos componentes do sistema, enquanto aqueles da categoria Comportamental lidam com a forma de interação de um componente com os demais do sistema. O padrão *Repulsion* é um exemplo de padrão Estrutural e o padrão *Evaporation* é um exemplo de padrão Comportamental.

O grupo Granularidade se refere ao escopo de aplicação das soluções e ele é dividido em 3 níveis: nível de sistema, nível de subsistema e nível de agente. O padrão arquitetural *Spreading* é um exemplo de solução a nível de sistema, pois é aplicado sobre todo o sistema; o padrão *Aggregation* é um exemplo de solução a nível de subsistema, já que ele é aplicado a partes do sistema; e o padrão *Replication* é um exemplo de solução a nível de agente, pois atua somente na arquitetura individual do agente.

O espaço de projeto SO:DuSE disponibiliza *Spreading, Gradient, Digital Pheromone, Repulsion, Flocking, Gossip* e *Quorum Sensing*. Eles estão situados no contexto de: Inspiração natural, Abstração concreta, Foco estrutural padrões arquiteturais e comportamental e Granularidade a nível de sistema, subsistema e agente. Os padrões arquiteturais citados podem estar interconectados, por sobreposição das funções ou serem combinados para alcance de algum objetivo ou requisito prioritário do sistema.

Os padrões arquiteturais *Spreading, Gradient* e *Digital Pheromone* são aplicados no SO:DuSE como Técnicas para Coordenação de Comunicação. O padrão *Spreading* é responsável por disseminar informações de um Agente para sua vizinhança. O padrão *Gradient*, agrega informações relevantes ao sistema e as dissemina em uma vizinhança, que por sua vez comunica as atualizações para outras vizinhanças. Desta forma, ondas de comunicação são emitidas, podendo serem direcionadas conforme abordagem ou necessidade que originou a comunicação. O padrão *Gradient* também pode ser encontrado, na literatura, com o nome Co-*Field*s (*coordination-fields*). Co-*Field*s são estruturas de dados distribuídas caracterizadas por um identificador único, um valor numérico de *location-dependent* e por uma regra de propagação que identifica qual o *field* deve ser distribuído na rede e como esses valores podem ser alterados durante a distribuição. Os Co-*Field*s requerem infraestrutura de suporte às suas variáveis (estruturas de dados) nos *hosts* que participem do ambiente no qual está inserido o sistema. No entanto, esta abordagem evita ao máximo o uso de algoritmos

complexos em sua representação e construção. Já o padrão *Digital Pheromone* é responsável por realizar comunicação indireta entre os agentes, quando, em seu mecanismo, ele possibilita agregar e dispor informações em um dado local do ambiente (alocação) para atualizar as informações dos Agentes que se relacionarem de alguma forma com aquele agente de infraestrutura [3].

Os padrões arquiteturais *Repulsion* e *Flocking* são aplicados no SO:DuSE como Técnicas para Coordenação de Movimentos. O padrão *Repulsion* é responsável por dispor uniformemente os agentes do sistema sobre uma área. Isso ocorre com o objetivo de evitar conflitos entre os agentes que possam gerar algum tipo de perda à operação dos serviços do sistema. O padrão *Flocking* utiliza os mecanismos básicos do padrão *Repulsion* para manter a distância desejável entre os agentes que se locomovem em uma formação que deve ser mantida até dado ponto do ambiente [3].

Os padrões arquiteturais *Gossip* e *Quorum Sensing* são aplicados no SO:DuSE como Técnicas para Coordenação de Tomada de Decisão. O padrão *Gossip* é responsável por realizar consenso entre todos os agentes do sistema ou subsistema. O padrão *Quorum Sensing* realiza eleição de líder, entre os agentes, considerando a densidade de agentes do sistema ou subsistema [3].

O trabalho [18] também exibe um gráfico que contém os padrões que se associam de alguma forma. Nele, os padrões são divididos em 4 conjuntos: padrões orientados a objetos, padrões bio-inspirados, padrões de grupos e padrões fechados. Os padrões arquiteturais reunidos no SO:DuSE estão inseridos na categoria de padrões arquiteturais bio-inspirados. Eles relacionam-se com alguns dos padrões dos demais conjuntos e internamente permitem os seguintes exemplos de interações: *Spreading-Repulsion*, *Gradient-Gossip*, *Digital Pheromone-Flocking*, *Spreading-Quorum Sensing*, entre outros. Essas relações acontecem porque eles podem compartilhar seus mecanismos e até participar de um mesmo processo dentro do sistema.

III. A Abordagem Proposta

Em cada uma das dimensões disponibilizadas no projeto de espaço SO:DuSE: Infraestrutura de Comunicação, Técnica de Comunicação, Técnica de Coordenação de Movimentos e Técnica de Coordenação de Tomada de Decisão, são encontradas soluções específicas, classificadas como pontos de variação. A dimensão Infraestrutura de Comunicação, por exemplo, apresenta os seguintes pontos de variação: Troca de Mensagens e Chamada Remota. Os pontos de variação são soluções complementares à arquitetura inicial proposta pelo SO:DuSE. Eles devem ser combinados para corresponder às características requisitadas nos sistemas auto-organizáveis.

As dimensões de projeto citadas foram traçadas em cima de uma arquitetura inicial constituída por 2 componentes: Agente e Ambiente. O componente `Agente` cumpre o papel de cliente nesta relação, enquanto o componente `Ambiente` faz as vezes do *Host* – hospedeiro, um componente físico do sistema no qual está implantando um agente virtual, ou Agente de Infraestrutura, conforme modelo computacional bio-inspirado. Juntos eles atuam para perceber o ambiente e realizar ações sobre a vizinhança ou sobre todo o sistema, de forma a modificar o cenário atual do sistema.

Conforme apresenta a figura 2: o componente **Agente** é inicialmente composto pelos elementos: `Conhecimento Corrente` – uma base de dados que mantém o estado corrente do componente – e `Percepção` – um componente responsável por trocar informações a respeito do conjunto de visões do ambiente denominada *foci* ou representação. O elemento Conhecimento Corrente disponibiliza duas interfaces: *Read-Write* e *Update*. O componente Percepção ainda disponibiliza as interfaces *Request* e *Sense*, sendo Sense uma interface direcionada para comunicação com componentes externos. O componente Percepção compartilha a interface *Update* com o componente Percepção para realizar troca de informações entre eles.

O componente **Ambiente** contém os elementos: `Gestor de Percepção`, responsável por manter o foco de percepção; `Estado`, base de dados responsável por guardar os dados das transações internas entre os componentes mais recentes; `Dinâmicas`, responsável por gerir os eventos arbitrários ao ambiente interno e ocorrências ou requisições dos agentes; e `Sincronização` responsável por suprir a base de dados Estado com a visão mais atual do ambiente externo. Para tanto, o elemento Sincronização tem duas responsabilidades: ler periodicamente o estado do ambiente externo e atualizar a base de dados Estado do Ambiente interno (virtual local) de acordo com o estado do ambiente externo. Os elementos Gestor de Percepção e Estado compartilham a interface *Read-Write*, pela qual se comunicam. O elemento Estado disponibiliza também a interface Update, utilizada para comunicação com os demais elementos do componente Ambiente, a exemplo dos elementos Dinâmicas e Sincronização. O elemento Gestor de Percepção compartilha a interface *Sense* com o elemento `Percepção` do Componente **Agente**. Ele ainda disponibiliza a interface *Observe* para comunicação com componentes externos, assim como o elemento Sincronização o faz com a interface *Synchronize*. Por fim, o conector **Shared Memory** liga os dois componentes: Agente e Ambiente, acoplando-se à interface Sense das portas relacionadas aos elementos Percepção e Gestor de Percepção.

A dimensão de projeto **`Infraestrutura de Comunicação`**, figuras 3 e 15, apresenta dois pontos de variação, como já citado: Troca de Mensagens e Chamada Remota. Ambos os pontos de variação desta dimensão adicionam os seguintes elementos à arquitetura inicial do sistema: `Comunicação`, no componente **Agente** e `Gestor de Comunicação`, no componente **Ambiente**. Os elementos Comunicação e Gestor de comunicação são submódulos dos respectivos componentes Agente e Ambiente responsáveis por gerenciar a troca de informações entre os agentes. A cardinalidade da relação entre os agentes é de 1 para N e eles podem se comunicar de maneira direta, quando em vizinhança (local) ou por comunicação indireta, quando em nível de subsistema (grupos). Os

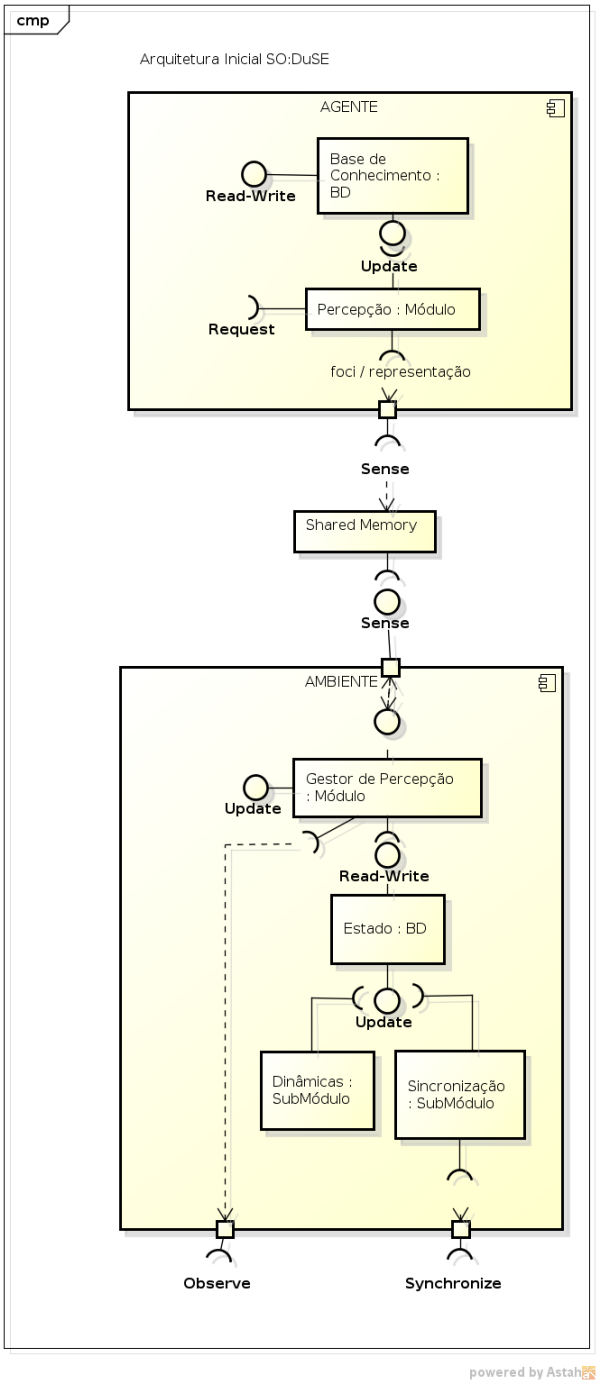

Figura 2. Arquitetura Inicial proposta para o SO:DuSE

dois componentes compartilham a interface *Send-Receive* para realizar atividades como: a coleta de mensagens, provimento de infraestrutura necessária para enviar e receber mensagens e direcionamento das mensagens aos agentes vizinhos. O elemento Comunicação interage com o repositório `Conhecimento Corrente`, do componente **Agente**, através da interface *Read-Write* disponibilizada pelo repositório. Ele, ainda, faz requisições ao elemento `Percepção` de igual componente, por compartilhar a interface *Request*. O elemento `Gestor de Comunicação` utiliza a interface *Read-Write* com o repositório Estado, do componente Ambiente. Ele ainda disponibiliza a interface *Transmit-Deliver* ao ambiente externo.

Nesta dimensão os componentes Agente e Ambiente relacionam-se através dos submódulos `Comunicação` e `Gestor de Comunicação`. No ponto de variação Troca de Mensagens, o conector ***Event*** é acoplado ao cenário, enquanto que no ponto de variação Chamada Remota, o conector **RPC** (*Remote Procedure Call*) é utilizado.

A dimensão de projeto **`Técnica de Comunicação`**, figuras 4, 17 e 18, apresenta os pontos de variação: *Spreading*, *Gradient* e *Digital Pheromone*. Todos os pontos de variação incluem os componentes Tomada de Decisão e Gestor de Ações nos componentes Agente e Ambiente, respectivamente. Eles interagem através da interface compartilhada *Act*, pela qual o componente Agente envia uma ação ao componente Ambiente. O módulo `Tomada de Decisão`, do componente **Agente**, utiliza a interface *Read-Write* disponibilizada pelo repositório `Conhecimento Corrente`. Ele também envia requisições ao módulo `Percepção`, através da interface *Request*. O elemento `Gestor de Ações`, localizado no componente **Ambiente**, se relaciona com o repositório `Estado` através da interface *Read-Write* e disponibiliza a interface *Operate* ao ambiente externo.

O ponto de variação *Spreading* apresenta o módulo Tomada de Decisão, do componente Agente. Ele contém os submódulos Controlador de Ação, Checador de Informações, Operador de Seleção, Operador de Refinamento e Gestor de Conflitos, que juntos são responsáveis pelo processo de *Spreading* sobre o sistema. Atuando sobre a Infraestrutura de Comunicação, a técnica de *Spreading* realiza *broadcast* de informações aos agentes da vizinhança do agente originador da ação. Dentre os submódulos do módulo em questão, o Checkador de Informações fica responsável por filtrar as informações recebidas e assim direcionar a função do Operador de Seleção.

Os pontos de variação *Gradient* e *Digital Pheromone* apresentam o módulo Tomada de Decisão, do componente Agente, contendo um esquema baseado em *field*. Essa estrutura é constituída basicamente por: um repositório Cache de Field e os submódulos: Checkador e Aggregador de Informações, Seletor de Ação, Controlador de Conflitos e Atualizador de Field. O ponto de variação Digital Pheronome difere do esquema baseado em field citado apenas por um submódulo, o Atribuidor de Prazo, que cumpre a função de um outro padrão arquitetural, o *Evaporation*. O padrão Evaporation faz com que o aspecto temporal seja implementado no sistema e desta forma possibilita priorizar as informações mais atuais ao sistema, tendo como base que elas são mais relevantes ao contexto de um ambiente altamente dinâmico. A solução *Gradient* é eficaz promove a agregação e disseminação de informações. Já a solução *Digital Pheromone* implementa uma solução que referente à comunicação indireta, no qual um valor é inserido em dado local do sistema para que seja lido por todos os agentes que por ele passe.

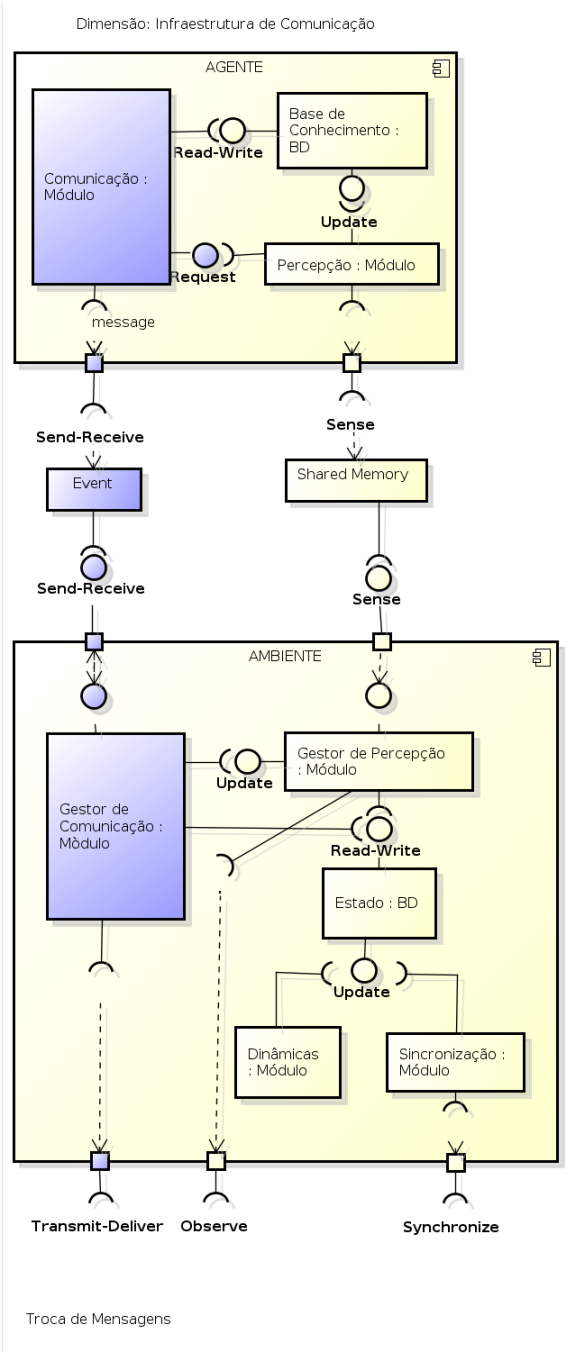

Figura 3. Dimensão Infraestrutura de Comunicação

Tomando como base a dimensão de Técnica de Comunicação, a qual divide suas soluções em duas categorias: spreading e baseada em *field*, as demais dimensões: Técnica para a Coordenação de Movimentos e Técnica para a Tomada de Decisão, são também segregadas para atender à essas duas estruturas distintas. A dimensão **`Técnica para Coordenação de Movimentos`**, figuras 5 20, 21 e 22, apresenta 4 pontos de variação, sendo eles: *Repulsion* baseado em *Spreading*, *Repulsion* baseado em *Field*, *Flocking* baseado em *Spreading* e *Flocking* baseado em *Field*. Ambos apresentam modificações

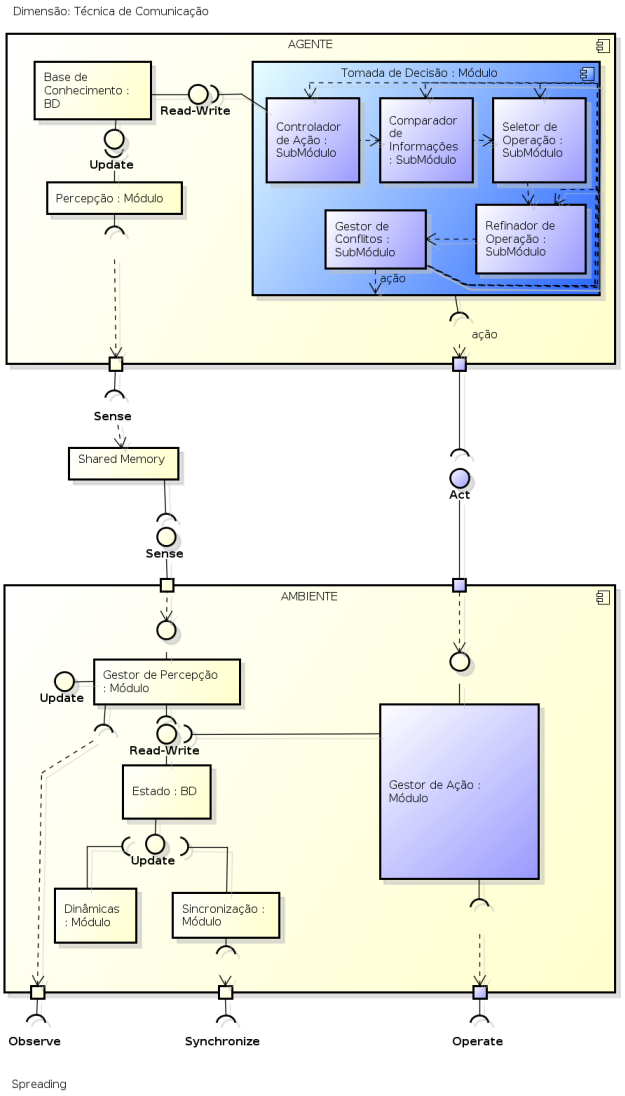

Figura 4. Dimensão de projeto Técnica de Comunicação

nos módulos Tomador de Decisão e Gestor de Ações referentes às soluções da dimensão Técnica de Comunicação.

O ponto de variação *Repulsion* baseado em *Spreading*, da dimensão Técnica para Coordenação de Movimentos acrescenta o módulo Roteamento e Cálculo de Posição, em relação à solução *Spreading*, da dimensão Técnica de Comunicação. Esse módulo é responsável por coletar e filtrar as informações relacionadas à localização do agente, em relação à sua vizinhança e calcular uma nova posição para ele baseada na uniformidade de cobertura de uma área física. Já o ponto de variação *Repulsion* Baseado em *Field* necessita incluir dois submódulos: Roteador e Calculador de *Field*, para cumprir iguais funções, tendo como base a abordagem de *field*, seja pela sobreposição da arquitetura do padrão *Gradient* ou do padrão *Digital Pheromone*.

Os pontos de variação *Flocking* baseado em *Spreading* e *Flocking* Baseado em *Field*, sobrepõem em apenas um

submódulo o desenho do ponto de variação *Repulsion*. O Regulador de Formação é responsável por manter um padrão de localização física dos agentes, enquanto eles se movimentam. O padrão *Flocking* atua no sistema para manter a formação dos grupos de agentes. Um exemplo de aplicação deste padrão está presente na atuação dos Kilobots, que são grupos de robôs miniaturizados capazes de cooperar para alcançar uma nova formação ou um novo destino, sem que haja a intervenção humana ou a centralização da tomada de decisão para tanto [19], [20].

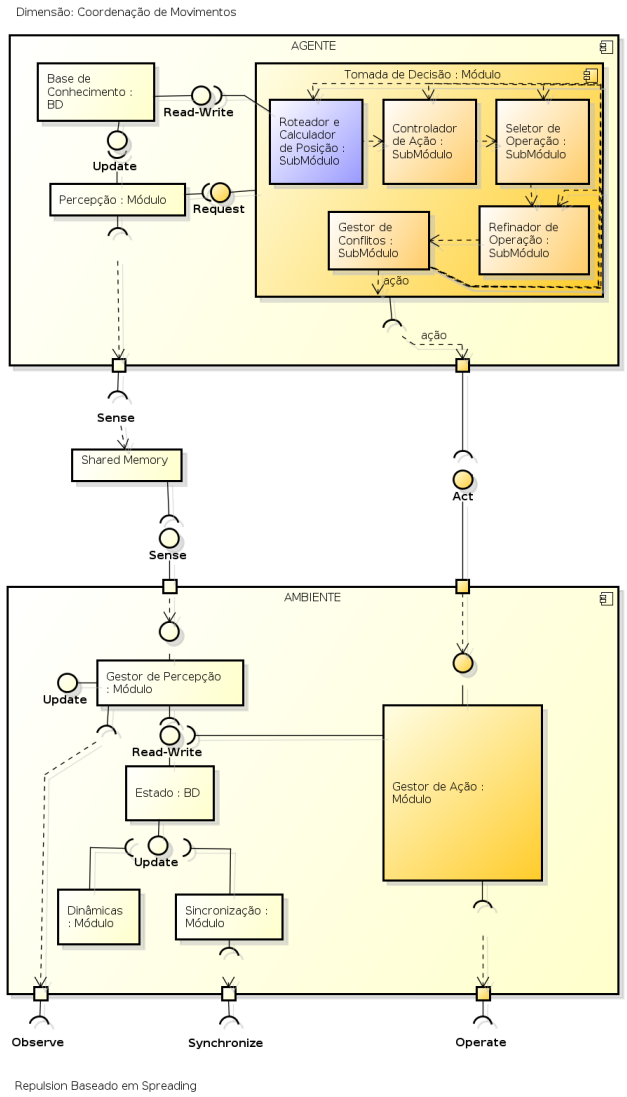

Figura 5. Dimensão de projeto Técnica para Coordenação de Movimentos

A dimensão de projeto **`Técnica para a Coordenação de Tomada de Decisão`**, figuras 6 24, 25 e 26, disponibiliza os pontos de variação: *Gossip* baseado em *Spreading*, *Gossip* baseado em *Field*, *Quorum Sensing* baseado em *Spreading* e *Quorum Sensing* baseado em *Field*. O ponto de variação *Gossip* apresenta, arquiteturalmente, o modelo do componente Tomador de Decisão, do componente Agente, igual à solução *Gradient* da dimensão Técnica de Comunicação. No entanto, elas diferem na implementação do componente Ambiente, já que a solução *Gossip* apresenta um esquema contendo o repositório Estado Local e Atualizador de Dados, somente para ele. O módulo Gestor de Ações recebe uma ação do componente Agente, atualiza os dados de informação local e envia uma requisição de atualização ao repositório Estado do componente Ambiente. O padrão arquitetural *Gossip* permite a tomada de decisão por "eleição de líder"após comparação de informações entre os agentes.

O ponto de variação *Gossip* baseado em *Spreading* segue modelo da solução *Spreading* da dimensão Técnica de Comunicação, enquanto que o ponto de variação *Gossip* baseado em *Field* implementa aquele modelo referente às soluções *Gradient* ou *Digital Pheromone*.

O ponto de variação *Quorum Sensing* inclui, em relação à solução arquitetural do *Gossip*, o submódulo Calculador de Densidade, no módulo Tomador de Decisão do componente Agente. Em sua implementação, à nível de código-fonte, ele necessitará conter uma função responsável por decidir se a proposta é aceita pela maioria dos agentes da vizinhança. Esse aspecto é comum às duas formas de variação: baseado em *Spreading* e baseado em *Field*. Esse ponto de variação, ainda, elimina a estrutura definida no ponto de variação *Gossip* para o componente Ambiente. O padrão *Quorum Sensing* implementa a tomada de decisão por consenso parametrizado por nível de densidade dos agentes atuantes no sistema.

2) Métricas de Qualidade: Foram definidas as seguintes métricas de qualidade que servirão de parâmetro para a validação da proposta apresentada neste trabalho: **escalabilidade** dos serviços, **sobrecarga de comunicação** entre os componentes (*overhead*) e **complexidade** arquitetural. A característica de escalabilidade representa a capacidade de reação do sistema ao aumento de demanda por seus serviços. Elasticidade e disponibilidade são indicativos de um sistema escalável. Medidas como o uso de servidores em nuvem, integração de *pool de threads* e replicação de dados ou dispositivos de *hardware* que ofertam os serviços auxiliam na implementação da característica de escalabilidade nos sistemas computacionais.

A característica de sobrecarga de comunicação está atribuída aos mecanismos de troca de informações que os componentes de um sistema utilizam para implementar e prover serviços. A depender da infraestrutura do sistema e da rede, arquitetura do sistema e técnica de comunicação que seja utilizada para suprir essa questão, o sistema pode apresentar pouco ou grande volume de informações trafegadas. É interessante disponibilizar medidas como: a aplicação do conector adequado para integrar os componentes do sistema, o uso de protocolo de comunicação em rede eficiente, aplicar normalização à base de dados integrada ao sistema e evitar a redundância de comunicação, para desta forma não sobrecarregar as funções de comunicação dos componentes do sistema, sua infraestrutura ou rede.

A característica de complexidade arquitetural está relacionada à reunião de todos os demais fatores que influenciam na qualidade e assertividade de um projeto arquitetural. Seu

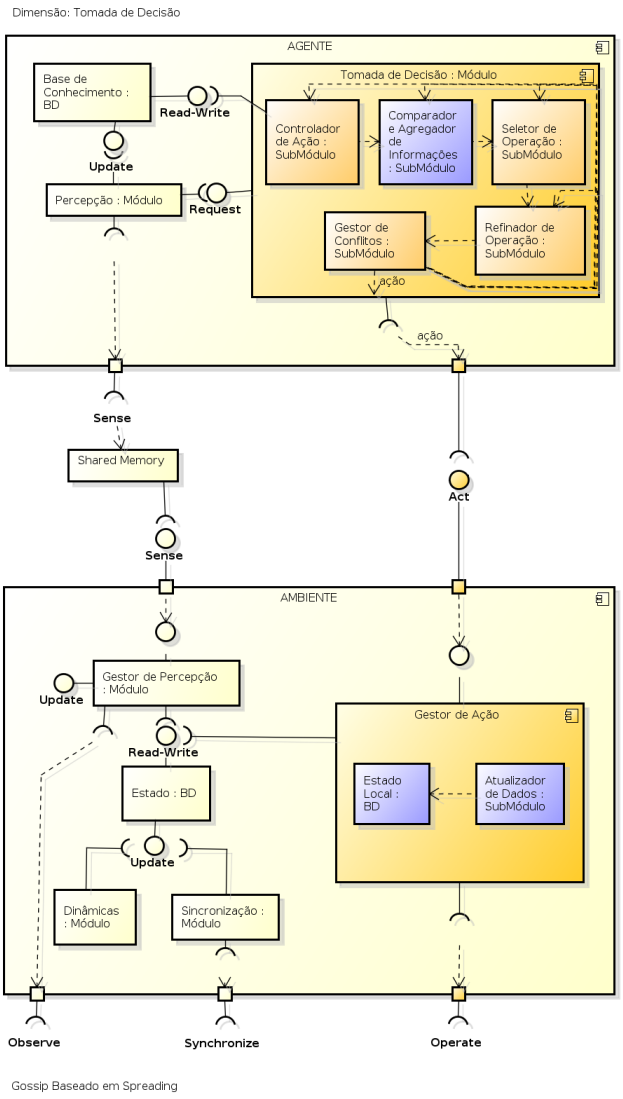

Figura 6. Dimensão de projeto Técnica para a Tomada de Decisão

conceito está bastante relacionado à maturidade das soluções arquiteturais aplicadas ao projeto. Outros fatores que influenciam na equação de complexidade arquitetural são: grau de manutenibilidade, escalabilidade, portabilidade, capacidade de reuso, mecanismos de comunicação e segurança e assim por diante. Ademais, fatores que aumentam sobremaneira a complexidade arquitetural dos sistemas auto-organizáveis são: mecanismo de coordenação descentralizado, tipo do processamento <*runtime*>, abertura ao acoplamento de novos componentes e configuração dos aspectos self-*.

A técnica de *Component-Point* (CP) foi aplicada à métrica de qualidade complexidade arquitetural para assim assim atestar a validade do espaço de projeto SO:DuSE. Esta técnica toma base no método de Ponto de Função para medir a complexidade arquitetural de um sistema especificado em diagrama UML. A técnica de *component-point* implementa as principais características da Análise de Pontos de Função sendo eles: a divisão de um sistema em diferentes tipos de componentes, diferentes critérios de dimensionamento para cada tipo de componente, o dimensionamento dos componentes, somatória do dimensionamento de todos os componentes. Isso resulta em possibilidade de extração do tamanho global da arquitetura e manipulação dos fatores de ajuste de contagem, se for o caso [21].

Neste trabalho foram definidos 2 *rankings* que permitiram conduzir a priorização das soluções arquiteturais auto-organizáveis catalogadas no SO:DuSE, conforme os parâmetros de escalabilidade e sobrecarga de comunicação. A tabela I apresenta essas duas listas, sendo que a coluna de escalabilidade está enumerada do maior para o menor e a coluna de sobrecarga de comunicação está enumerada de maneira inversa à anterior. Para priorizar as soluções conforme métrica de complexidade arquitetural foi utilizado um algoritmo à parte.

Tabela I
RANKING DAS SOLUÇÕES ARQUITETURAIS CONFORME ESCALABILIDADE OU SOBRECARGA DE COMUNICAÇÃO

Nº	Escalabilidade (decrescente)	*Overhead* (Crescente)
1	Event	RPC
2	RPC	Digital Pheromone
3	Spreading	Event
4	Digital Pheromone	Spreading
5	Gradient	Gradient
6	Gossip Spreading	Repulsion Field
7	Gossip Field	Flocking Field
8	Quorum Sensing Spreading	Gossip Field
9	Quorum Sensing Field	Quorum Sensing Field
10	Repulsion Spreading	Repulsion Spreading
11	Repulsion Field	Flocking Spreading
12	Flocking Spreading	Gossip Spreading
13	Flocking Field	Quorum Sensing Spreading

O resultado de validação do SO:DuSE apresentou três soluções arquiteturais de sistemas auto-organizáveis viáveis. Se esses *rankings* relatados fossem considerados neste trabalho, de forma isolada, para fim de uso sobre algoritmo de força-bruta, tais soluções provavelmente transitariam na parte superior da tabela I, onde são apresentadas soluções altamente escaláveis e de baixa sobrecarga de comunicação. No entanto, a aplicação da métrica de complexidade arquitetural, assim como técnica de otimização por busca, conferiram diversidade a esse resultado.

A. Aspectos de Implementação

Para concretizar a implementação do espaço de projeto SO:DuSE foi necessário criar dois arquivos XMI contendo informações a respeito: 1) da arquitetura inicial; e 2) das dimensões de projeto já citadas neste trabalho. Todos os aspectos de implementação via código-fonte, deste trabalho, tiveram como base os modelos arquiteturais descritos na seção II-E1.

O arquivo XMI contendo as informações a respeito da arquitetura inicial é o documento utilizado na entrada do processo de refatoração de modelos arquiteturais do SO:DuSE. Esse processo cumpre o formato genérico do processo DuSE

e seu modelo arquitetural de entrada, denominado arquitetura inicial, que deve apresentar as configurações básicas de um sistema. No estudo de caso aplicado neste trabalho, o modelo inicial proposto apresenta o escopo mínimo de um sistema auto-gerenciável. Sendo assim, suas configurações básicas são: os componentes `Agente` e `Ambiente`, que se comunicam a partir de um conector do tipo *communication*. Ambos componentes apresentam os módulos de percepção, dinâmicas e sincronização e banco de dados relacionados ao estado do ambiente interno e externo ao sistema, isso, para dotar os agentes com capacidade de sensorear e responder aos estímulos do ambiente. Nesta etapa da implementação foi utilizado o *plug-in* Papyrus vinculado à IDE Eclipse para gerar o arquivo XMI conforme o modelo componente-conector UML foi traçado.

O arquivo XMI contendo as informações a respeito das dimensões arquiteturais que um sistema auto-gerenciável deve ter é o documento utilizado para alterar o modelo de entrada no processo SO:DuSE. É ele quem disponibiliza os parâmetros que permitem conduzir a refatoração do modelo de arquitetura inicial em modelos auto-organizáveis. Cada uma das dimensões descritas no arquivo pode ser identificada como um novo pacote de elementos a ser inserido no arquivo XMI do modelo de saída, referente ao modelo inicial refatorado. Com isso, o novo sistema deve ao menos apresentar as características de interatividade inerentes a um contexto de operação sensível.

IV. Validação

Os resultados apresentados por este trabalho foram validados com vistas ao processo de otimização presente na ferramenta DuSE-MT. Essa ferramenta possibilita a escolha de soluções arquiteturais ótimas sobre um espaço de busca relacionado a um problema modelado para um espaço de projeto. No caso do SO:DuSE, deveriam ser encontradas soluções arquiteturais ótimas para o problema multiobjetivo da escolha de soluções arquiteturais auto-organizáveis em seu espaço de busca.

A. Estudo de Caso

O estudo de caso relacionado a este trabalho tem como contexto a coordenação de movimentos dos agentes que integram um sistema de controle logístico para almoxarifado industrial, [18]. Neste ambiente, existe demanda por transporte de materiais no estoque de queijos da fábrica. O estudo de caso compreende a comparação entre a arquitetura dos dois sistemas utilizados com o propósito de guiar a logística deste estoque de queijos: o primeiro, centralizado, e o segundo, descentralizado. Ambos os sistemas são compostos por veículos guiados automaticamente (AGVs) que precisam trabalhar juntos para transportar as cargas do estoque. Os fluxos de transporte são gerados pelos sistemas clientes, em específico ERPs(sigla para *enterprise resource planning*).

O sistema previamente instalado no estoque de queijos da fábrica trabalhada de maneira centralizada, o que facilitava, por exemplo, a detecção de falhas. O servidor guardava a função de planejar o escalonamento para todas as tarefas dos AGVs envolvidos, de atribuir a eles atividades e constantemente alterar seus *status*. Como resultado, o sistema era `confiável` e `previsível`. No entanto, uma mudança nos requisitos do sistema pôs em cheque o projeto de arquitetura utilizada pelo sistema, a arquitetura centralizada. Os clientes requisitaram capacidade de **auto-gerenciamento** e portanto, aspectos de qualidade como `flexibilidade` e `abertura` a novos componentes passaram a ser necessárias de adoção nesse sistema. Flexibilidade referente à habilidade de o sistema lidar com as condições dinâmicas de operação no sistema.

O novo sistema foi desenvolvido entre os anos de 2004 e 2007 por um time de engenheiros e desenvolvedores de sistemas da empresa Egemin e pesquisadores do laboratório DistriNet. A característica de auto-organização, requisito que motivou a construção do novo sistema, conferiu avanços ao controle da logística no estoque porque proporcionou ganho em agilidade e facilitação da resolução de conflitos na coordenação de movimentos dos agentes.

Para contemplar os novos requisitos especificados pelo cliente, a arquitetura do sistema precisou ser refatorada. Na realidade, ela foi projetada do zero, para substituir o sistema que operava até então. Desta forma, sua arquitetura passou a ser guiada pelas características de auto-gerenciamento, flexibilidade e abertura. Outros aspectos pertinentes aos sistemas descentralizados como tomada de decisão, eficiência e sobrecarga (tempo de resposta e consumo de largura de banda da rede) foram considerados também. Com a mudança de estrutura do sistema, os AGVs participantes ganharam a função de poderem adaptar a si próprios de acordo com a situação do ambiente e vizinhança. As validações do novo sistema ocorrem sobre implementação de protótipo e uso de AGVs reais. Ele foi testado através de simulações.

O novo sistema disponibiliza as seguintes funções: atribuir tarefas de transporte aos AGVs; rotear os caminhos dos AGVs eficientemente, no decorrer das trilhas; evitar colisões e *deadlocks* entre os AGVs; e recarregar a bateria entre os AGVs. O sistema também precisa lidar com a seguinte configuração de ambiente: interagir com ambiente `dinâmico` (pois o ambiente altera constantemente as condições de operação do sistema); o fluxo de entrega dos transportes (tarefas) é `irregular` e `imprevisível`. Apesar das inconstâncias e conflitos que ocorram, o sistema deve funcionar de maneira `eficiente` e `robusta`. Alguns eventos que podem ocorrer neste cenário são: os AGVs podem deixar e retornar ao sistema para manutenção (ex.: recarga de bateria); as máquinas de produção podem variar em tempo de espera; além de `distúrbios` como: obstrução de rotas do mapa por bloqueio de carga; interrupção ou sobrecarga do fornecimento de material; mudança ou fechamento de trilhas do mapa, entre outras.

Os esforços para projetar e desenvolver o novo sistema reuniram valor equivalente a 8 anos de homem/trabalho, tempo necessário para planejar, projetar e construir o sistema. A criação do SO:DuSE visa oferecer suporte a cenários como este, onde é necessária a recondução de sistemas centralizados, ditos convencionais, em sistemas descentralizados,

ditos abertos. Do trabalho que originou o estudo de caso, [18], se pode derivar uma arquitetura inicial para sistemas auto-organizáveis, constituída por 2 componentes arquiteturais genéricos: Ambiente (Virtual) e Agente.

B. Outro Exemplo de Aplicação

Os aspectos de projeto descritos neste trabalho também podem ser aplicados em projetos de sistemas com domínio em situações de crise em grupo, onde se faz necessária a coordenação do movimento das pessoas de uma área de risco para uma área considerada segura. Para tanto, os indivíduos participantes do grupo devem ser portadores de dispositivos móveis ou qualquer outro equipamento eletrônico dotado de capacidade sensorial, rede ativa e autonomia de operação. O fundamento deste exemplo de aplicação dos sistemas auto-organizáveis se encontra integralmente nos seguintes trabalhos: [16], [22], [23], [24]. Eles apresentam o tema citado no âmbito da visita de turistas a um museu e no ambiente virtual do jogo Quake 3 Arena. O museu propicia ambiente pervasivo aos visitantes e possibilita, inclusive, o processo de evacuação dos turistas em caso de acidente ou perigo iminente no local. O jogo ganhou o suporte da ubiquidade para a interação mais dinâmica e natural entre os robôs virtuais, quando da formação e iminência de ataques.

Este outro cenário proposto neste trabalho, que toma como base no contexto citado, tem aplicação na coordenação de movimento das pessoas agrupadas em local fechado, quando da necessidade de evacuação por motivo de emergência. Para tanto deve existir neste cenário: (i) um local propício ao agrupamento de pessoas, aqui definido como um *shopping center*; (ii) ambiente equipado com infraestrutura adequada à comunicação em rede; (*iii*) a presença de dispositivos móveis ativos dotados de capacidade de interação com o ambiente. Os dispositivos que participam deste ambiente devem utilizar a rede disponibilizada pelo *shopping center* com motivo de parear seus dispositivos com os demais ativos no local. Assim, a rede, apesar de central, possibilita a comunicação e interação entre os dispositivos, que passam a orientar-se mutuamente para a tomada de decisão.

O objetivo em apresentar este cenário é avaliar as possibilidades de construção de um sistema auto-gerenciável que o suporte. Ainda, seu objetivo específico é analisar as combinações e soluções arquiteturais possíveis pautadas em múltiplas combinações capazes de solucionar ou suportar os problemas de tomada de decisão com foco na coordenação de movimentos em grupo. O resultado do alcance desses objetivos foi a extração do modelo inicial passível de ser utilizado no SO:DuSE para refatoração arquitetural.

As abordagens CO-*Field*s e EFA poderiam utilizadas na correlação entre os problemas de decisão encontrados no momento da coordenação de movimentos para evacuação por emergência e soluções propostas. Ambos poderiam ser aplicadas aos limiares dos padrões arquiteturais condizentes com a necessidade do problema. E então seriam comparadas como o intuito de extrair opções de soluções ao problema da coordenação de agentes em grupo. O universo de apoio à construção das soluções para os problemas de tomada de decisão acerca deste estudo de caso está restrito às metáforas **auto-organizáveis**: *Foraging, Molding, Embryogenesis, Brood Sorting, Flocking/Schooling and Herding* e *Quorum* [25]. Portanto, neste estudo de caso, podem ser avaliados os resultados obtidos com o uso dos padrões: *Gradient*, *Flocking* e Morphogenesis, relacionados à uma ou mais das metáforas citadas anteriormente.

A proposta de uso dos CO-*Field*s é representada no ambiente virtual do jogo Quake III Arena [22] com o objetivo de prover dinamicamente os robôs com mecanismos de interação efetiva, simples, fácil de ser obtida e expressar informação contextual. O uso de Co-*Field*s sobre o jogo Quake III Arena permite: o reconhecimento do local e dos vizinhos pelos robôs, obstrução de elementos de escape (fechamento das portas) e distância regular na manutenção do movimento estabelecido entre eles (formação regular - flocking). Assim, é possível levar os robôs a conhecerem o local, distribuí-los de acordo com os padrões espaciais, cercar um inimigo ou simplesmente movê-los no ambiente sem interferência dos demais. A inspiração desta teoria surgiu da forma com que as partículas se movem e se auto-organizam globalmente no mundo físico real. Elas se movem de acordo com a informação contextual que é representada por *potential fields*. Nessa estrutura, o ambiente e a informação contextual são representadas na forma de *computational fields* distribuídos (Co-*Field*s).

No problema de coordenação de robôs virtuais do jogo Quake III Arena estão presentes conceitos importantes ao tema da locomoção em grupo. São os seguintes *bulding blocks:* (*i*) Mecamismo de interação – os robôs precisam se comunicar para planejar, decidir e sincronizar suas ações; e (*ii*) *Context Awareness* – a sincronização de ações deve ser mais proveitosa se acontecer baseado no conhecimento dos agentes que estão em volta.

Eletric Field Approach (EFA) é uma abordagem difundida no campo da robótica e pode ser usada para controlar um time de robôs. Neste cenário, cada robô obtém as informações do ambiente através da captura e armazenamento de imagens. As decisões sobre os movimentos são tomadas pelo exame dos *gradient fields* presentes nestas capturas. EFA e CO-*Field*s diferem no aspecto de implementação. Co-*Field*s são estruturas de dados distribuídas espalhadas no ambiente; enquanto no EFA, os *fields* são apenas representações de um passado recente individuais e internas aos agentes.

A figura 7 exibe o modelo de formação do ambiente quando padrão arquitetural *flocking* é aplicado sobre a estrutura do ambiente do problema (museu ou jogo). Nessa condição, cada agente do sistema é responsável por criar seu próprio *flock field*. Ele deve conter a informação da distância mínima, 8, a ser preservada na interação dele com os demais robôs do ambiente, para manter a formação entre eles. Essa distância deve crescer incrementalmente.

Cada agente do sistema deve avaliar sua coordenada, compará-la com um mínimo de combinação de todos os outros agentes (vizinhança) e simplesmente repelir as demais coordenadas obtidas. Globalmente o sistema se auto-organiza em uma

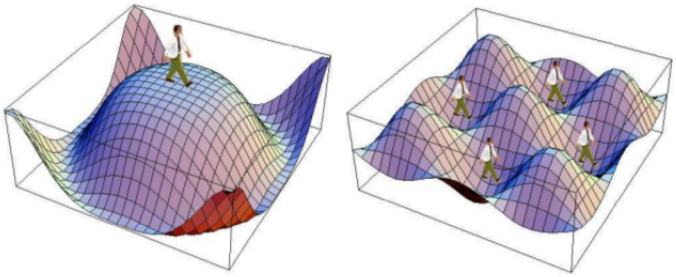

Figura 7. Formação regular com o padrão *Flocking* [23]

Equação da distância:
CF(x, y, t) = min(FLOCKi(x, y, t) : i = 1, 2, . . . , n)
] Onde: Flock = flock field | i = agente | CF =

coordination field.

Figura 8. Representação matemática do padrão *Flocking*

grid quase regular, porque todos os agentes tentam permanecer em um dos pontos dos campos de distância mínima dos demais agentes e assim mantém uma formação regular.

No plano de evacuação por emergência, em um museu tecnologicamente equipado para integrar os visitantes com atendimento virtual especializado, as paredes do museu devem abrigar a infraestrutura computacional embutida necessária ao funcionamento do plano de interação ubíqua. Essa seria a forma de ligar os visitantes à estrutura do ambiente. Os *hosts* ativos no ambiente funcionariam então como *middleware* nesta situação, por prover o suporte ao armazenamento dos dados da comunicação(armazenar valores dos *fields*) e os próprios mecanismos de comunicação(propagar os *fields*).

Os dispositivos móveis dos visitantes devem ser acoplados dinamicamente a esta rede, precisando apenas que eles estejam supridos de agentes de *software* operando algum tipo de rede *wireless* para usufruir deste cenário. Quando acoplados à estrutura de *hosts* embutidos, eles são capazes de disseminar *fields* no ambiente, assim como ler os valores dos *fields* presentes na vizinhança. Desta forma é possível coordenar o movimento dos visitantes no local do museu e assim os dirigir a um local seguro, em caso de crise ou catástrofe.

Este exemplo de aplicação dos sistemas auto-organizáveis apresenta uma adaptação da visão de coordenação de movimentos no museu ubíquo citado e da visão de coordenação de movimentos dos robôs virtuais no jogo Quake III Arena. Ele foi aplicado, no estudo de caso com visão à situação de coordenação de movimentos de pessoas por necessidade de evacuação no momento de crise, com a diferença de que os indivíduos devem repelir a coordenada da crise (*Repulsion*), mas agruparem-se em um local seguro (*Flocking*).

A figura 9 apresenta os passos sequenciais de aplicação deste exemplo. O cenário se inicia com a comunicação centralizada na rede disponibilizada pelo ambiente. Com a ocorrência da crise, sinais unilaterais do ambiente e bilaterais dos agentes passam a ocorrer sobre o ambiente. Essa é a forma com que os agentes tomam "consciência"do ocorrido e consultam seus vizinhos para tomar decisões. Com a disseminação dos *gradients* no ambiente e estabelecimento da comunicação entre os agentes, uma rede temporária, interna ao grupo, é formada no ambiente e os agentes passam a cooperar para alcançar o local seguro, conforme exibe o último passo da figura.

A figura 10 exibe a representação de projeto arquitetural do relacionamento entre as entidades Agente e Ambiente sobre o sistema auto-organizável. Ela se adéqua ao modelo de arquitetura Agente-Ambiente inseridos na proposta de implementação deste trabalho, evidenciando que ele é de fato um modelo que representa um modelo de entrada ideal ao SO:DuSE pois apresenta elementos básicos de sistemas candidatos à refatoração por meio deste processo.

Do modelo básico derivado do atual estudo de caso, a entidade Ambiente pode se relacionar com vários agentes, o inverso também é possível no contexto real, mas não será considerado no cenário aqui definido. A entidade Agente apresenta auto-relacionamento para explicitar que os agentes devem ser relacionar para cooperar. Desta forma, o sistema deve possuir ao menos 2 agentes se relacionando e também com o ambiente, para que assim o sistema possa ser considerado auto-organizável. A cooperação é o princípio básico deste tipo de sistema.

No cenário descrito, o relacionamento entre Ambiente e Agente, pode acontecer de três formas diferentes: através da aplicação de um dos seguintes padrões arquiteturais por vez: *Gradient, Repulsion, Flocking* ou *Morphogenesis*. Sendo assim, deste modelo podem ser derivados 4 soluções arquiteturais diferentes, que podem ser combinadas de diversas maneiras para conduzir o projeto a resultados distintos.

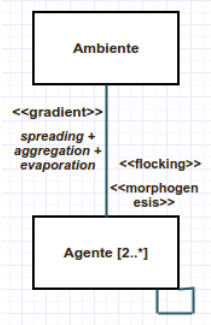

Figura 10. Relacionamento entre as entidades do sistema

C. Configuração do Experimento

O DuSE-MT utiliza a busca combinatorial multiobjetivo com a aplicação do algoritmo evolucionário NSGA-II para

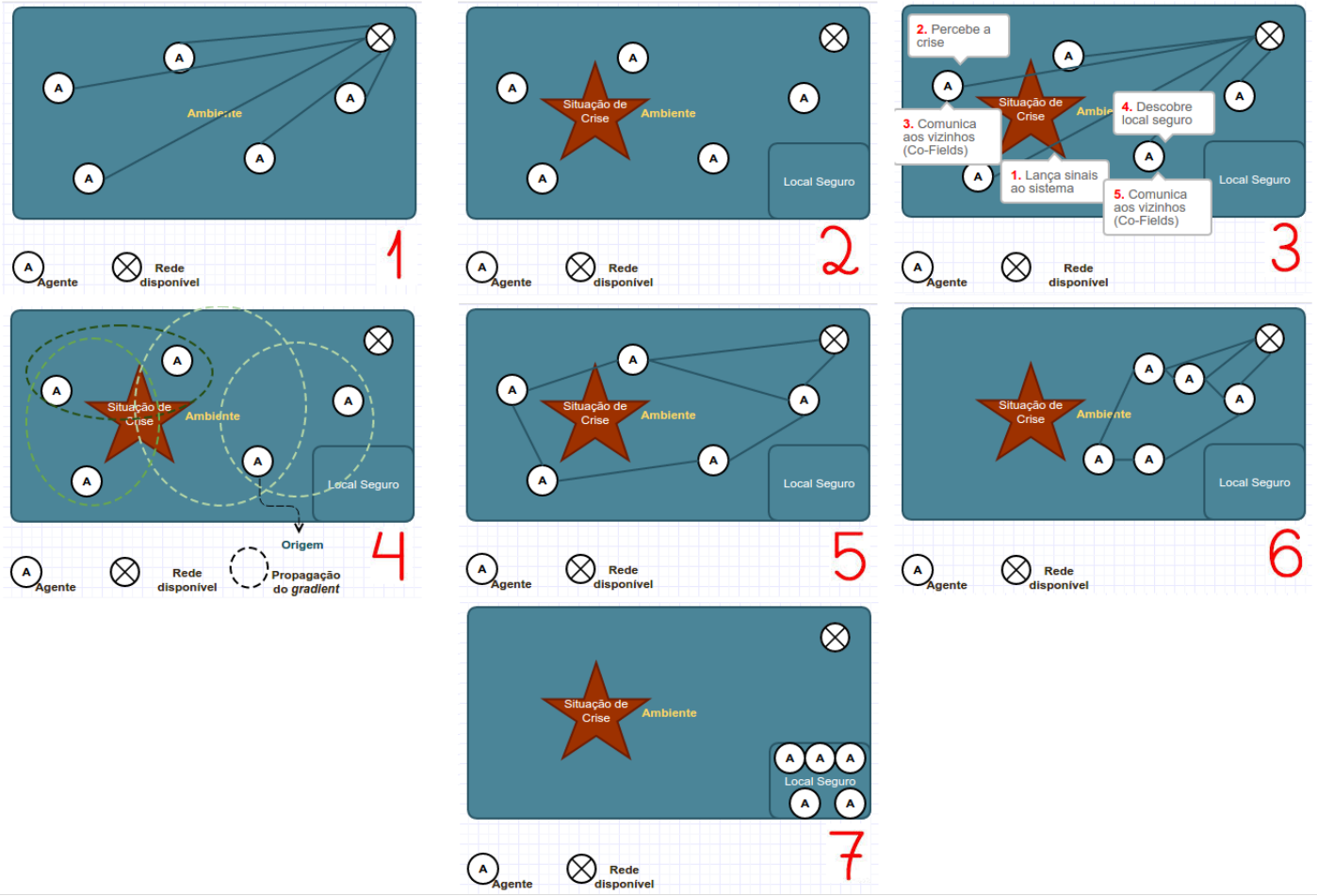

Figura 9. Exemplo de aplicação da abordagem CO-*Field*s em um sistema auto-organizável

refatorar modelos arquiteturais genéricos. Os algoritmos evolutivos costumam apresentar três operações básicas: 1 – Combinação (*Crosssover*); 2 – Mutação (*Mutation*); e 3 – Seleção (*Selection*). A operação de combinação cumpre a função de misturar características dos indivíduos das populações para gerar novos indivíduos. A operação de mutação visa acrescentar maior valor a dada característica, comum a alguns indivíduos da população, para com isso, diferenciá-los quando suas gerações ficam muito parecidas. A operação de seleção cumpre função de escolher as soluções mais adequadas às funções-objetivo do problema. Essas operações são aplicadas de maneira coordenada e múltiplas vezes sobre a populações, nas diversas gerações relacionadas às iterações de execução do algoritmo, para com isso propiciar a escolha das soluções que mais se pareçam com o resultado ideal das funções-objetivo.

Este trabalho propõe a solução do problema multiobjetivo de escolha das soluções arquiteturais para o espaço de projeto SO:DuSE, estabelecendo as seguintes funções-objetivo: 1 – maximizar escalabilidade; 2 – minimizar sobrecarga de comunicação e 3 – minimizar complexidade arquitetural(CP). A ferramenta DuSE-MT foi configurada para executar 300 iterações de otimização, com população de 10 indivíduos (treze arquiteturas), cada. Os operadores DuSECrossover e DuSE-MutationOper foram utilizados para sobrepor o operadores de combinação e mutação padrão do algoritmo NSGA-II e assim garantir, respectivamente, recombinação de 2 pontos com probabilidade de 0.9 e mutação bit-a-bit com probabilidade igual a 1. Isso foi realizado com o intuito de garantir a diversidade dos indivíduos, nas populações das iterações e diminuir a possibilidade de ocorrência de arquiteturas inválidas.

A validação do SO:DuSE foi conduzida pelo quesito de efetividade da captura de *tradeoffs* arquiteturais voltados aos sistemas auto-organizáveis. Como a ferramenta DuSE-MT utiliza o método de otimização multiobjetivo para realizar a seleção das soluções arquiteturais ótimas para refatorar o modelo arquitetural inicial, foi necessário detectar e avaliar as características de qualidade da frente de Pareto[2] do problema citado. Então, após executar 30 iterações de otimização para formar uma frente de Pareto de referência P*, foi realizada a

[2]Frente de Pareto ou *Pareto-front*: É um conceito de otimização multiobjetivo que visa garantir a solução ideal global, conforme equilíbrio das proporções atribuídas às variáveis de um problema. A solução ótima é garantida quando não há formas de se melhorar a situação de uma variável do problema sem que haja degradação de pelo menos uma das demais.

execução de mais 1 iteração para obter a frente de Pareto P, para comparação.

Uma frente de Pareto que apresente alta cardinalidade e espalhamento de soluções remete à efetividade do SO:DuSE em capturar *trade-offs* de relevância ao problema atacado. Desta forma, a execução de mais do que 31 execuções não apresentou alteração significativa na cardinalidade de soluções apontadas na frente de Pareto. Por fim, os resultados desta validação foram obtidos em uma máquina d processador Intel Core i7 com 8GB de memória RAM.

D. Resultados

Os resultados extraídos das execuções de otimização foram analisados com objetivo de verificar a aproximação de resultados entre **P** (frente de Pareto final) e **P*** (frente de Pareto de referência). No decorrer das 31 execuções de otimização, estima-se que foram verificadas 2976 arquiteturas das 6912 arquiteturas disponíveis no espaço de busca. Nesse valor estão provavelmente estão inclusas arquiteturas repetidas. Com isso, P* apresentou 3 arquiteturas candidatas das quais P deve se aproximar ao máximo.

Os testes foram exercitados em 2 grupos distintos de arquiteturas para sistemas do tipo auto-organizável. Os grupos de soluções *Spreading* e *Gradient* encontradas no exercício dos testes são apresentados nos gráficos de dispersão da figura 11. A matriz apresenta a comparação entre as soluções do P* e P, conforme demonstração das funções-objetivo 1, 2 e 3. As soluções exibidas na cor vermelha fazem parte do grupo *Spreading* e aquelas de cor azul pertencem ao grupo *Gradient*.

Na matriz, os gráficos presentes em sua linha diagonal são do tipo histograma e portanto apresentam valores referentes à frequência de ocorrência das soluções. Nos histogramas, as barras que apresentam hachuras se referem às arquiteturas presentes em P* enquanto que as barras que não apresentam hachuras se referem àquelas dominadas presentes em toda a população.

Os demais gráficos da matriz exibem as soluções encontradas para P e P* no espaço de busca. As soluções dispostas nos gráficos em formato de losango participam da frente de Pareto P e as outras presentes em formato de círculo são as soluções dominadas. O tamanho de cada losango ou círculo indica o grau de adaptabilidade do controlador de otimização, quanto maior, mais conflitante é a solução (efetividade).

O conteúdo das células: linha 2 e coluna 3 ou linha 3 e coluna 2, da matriz de gráficos de dispersão, nos permite verificar a correta captura da relação entre os *trade-offs* escalabilidade e sobrecarga de comunicação. Em ambos os gráficos, foram identificados 4 soluções arquiteturais na frente de Pareto P e soluções dominadas muito próximas também desta frente. Isso sinaliza que é possível obter alta escalabilidade e baixa sobrecarga de comunicação sobre sistemas auto-organizáveis com base nas treze arquiteturas disponibilizadas por este trabalho.

Ao observar a figura 11, nas células: linha 1 e coluna 3 ou linha 3 e coluna 1 também nos permite certificar que obter baixa complexidade arquitetural e baixa sobrecarga de comunicação é uma atividade possível. Os 2 gráficos exibidos nas células citadas apresentam 1 solução arquitetural viável dentre as treze arquiteturas e algumas soluções dominadas ficam próximas do limiar desejado (frente de Pareto P).

O conteúdo das células: linha 1 e coluna 2 ou linha 2 e coluna 2 nos apresentam gráficos contendo soluções dispersas no espaço de busca com algumas delas formando uma frente de Pareto parcial (losangos). Isso implica afirmar que é pouco viável obter um projeto arquitetural para sistema auto-organizável contendo alta escalabilidade e baixa complexidade arquitetural. Esse resultado, apesar de nos distanciar de um ideal de otimização, nos remete à eficácia das soluções arquiteturais apresentadas neste trabalho, já que é pouco confiável a um sistema possuir essas duas características ao mesmo tempo.

A métrica de qualidade escalabilidade [coluna 2 e linha 2] apresenta resultados efetivos e mais frequentes entre os valores de probabilidade 0.5 e 1. A métrica de qualidade sobrecarga de comunicação [coluna 3 e linha 3] apresenta resultados efetivos entre os valores de probabilidade 0 e 0.5, assim como as mais frequentes. Já a métrica de qualidade complexidade arquitetural [coluna 1 e linha 1] não apresenta resultados efetivos (presentes em uma frente de Pareto), mas exibe resultados frequentes entre os valores 0 e 35.

Existe *trade-off* característico na relação entre as métricas de qualidade escalabilidade e sobrecarga de comunicação. No gráfico que exibe esta relação 6 resultados efetivos na frente de Pareto são evidenciados. Sua frente de Pareto é a maior e mais espalhada em relação aos resultados exibidos nos demais gráficos. Isso indica que o problema de otimização relacionada a essas duas variáveis: maximizar escalabilidade e minimizar sobrecarga de comunicação é de fato multiobjetivo. A frente de Pareto entre as qualidades de sobrecarga de comunicação e complexidade arquitetural apresenta um único resultado efetivo em P, indicando que tais objetivos: minimizar sobrecarga de comunicação e minimizar complexidade arquitetural são pouco conflitantes. Por fim, a relação entre as qualidades escalabilidade e complexidade arquitetural não demonstra ser conflitante por definir apenas uma frente de Pareto pois podem ser verificadas, no gráfico relacionado, 5 soluções arquiteturais ótimas viáveis. Com isso, a conclusão a que podemos chegar é de que existe distribuição aparentemente normal para as soluções de alta escalabilidade e baixa sobrecarga de comunicação, o que indica que essas são soluções difíceis de encontrar por algoritmo de força bruta ou de forma manual.

A figura 12 apresenta como a métrica *hypervolume*[3] evolui ao longo das sucessivas execuções da otimização sobre o problema SO:DuSE. Cada uma das iterações de otimização apresentou um valor de *hypervolume* assim como aquela em que foi detectada a frente de Pareto P. Então, o objetivo desta fase de validação é verificar quais das demais iterações mais se aproximou de P em relação à sua métrica de *hypervolume*.

Na figura estão presentes os valores mínimo, médio e

[3]Hypervolume = métrica que calcula a área (para o caso de problemas com dois objetivos) ou o volume (para o caso de problemas com três ou mais objetivos) de encontro entre as dimensões de um problema em seu espaço de busca de soluções.

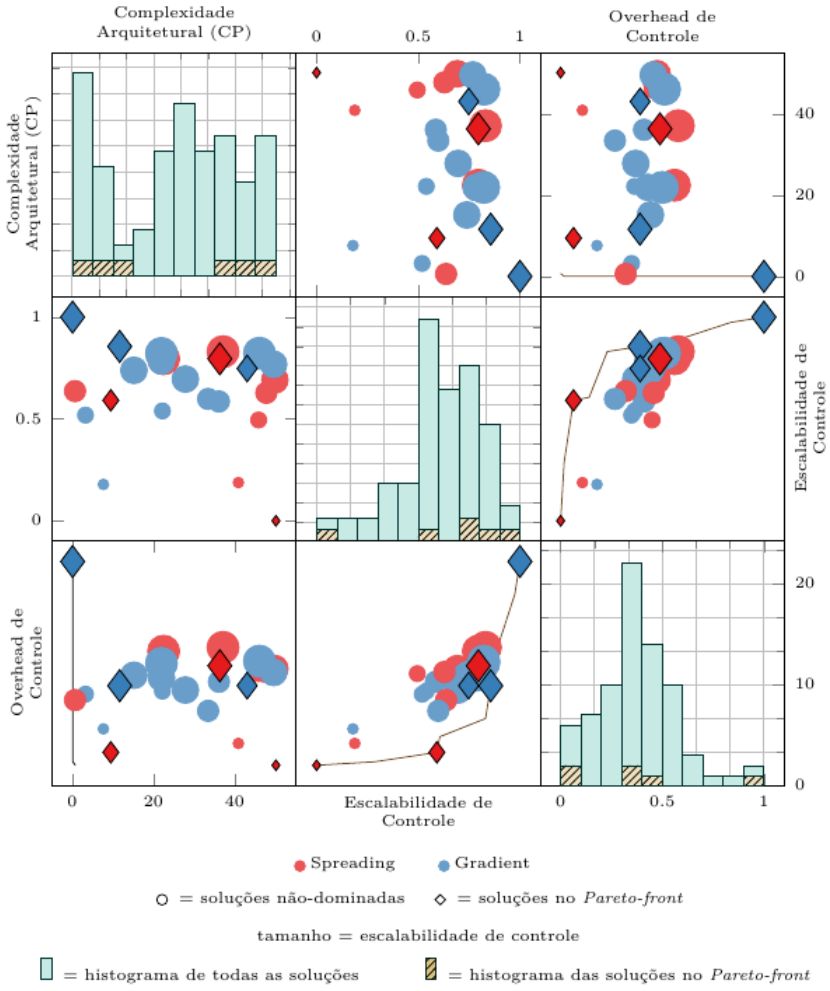

Figura 11. Matriz de gráficos de dispersão do P*

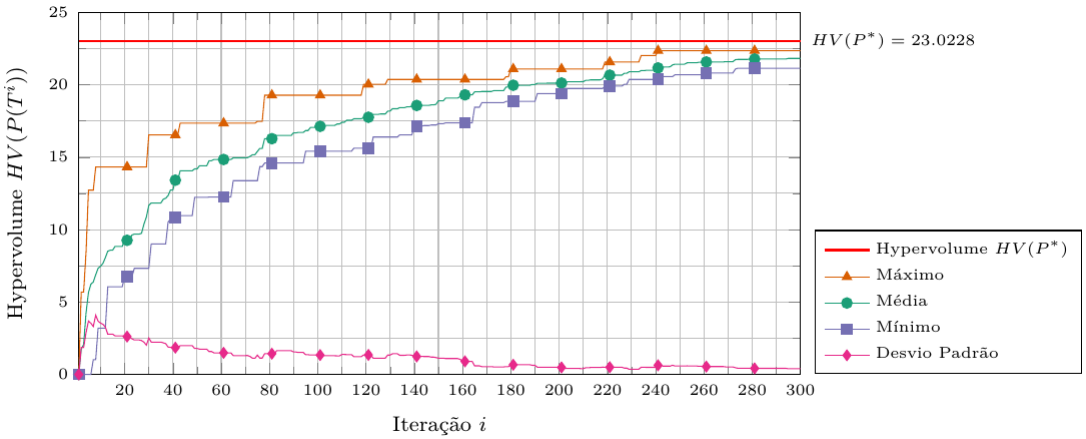

Figura 12. Gráfico de convergência da métrica *hypervolume* no decorrer a execução da otimização

máximo assim como o desvio padrão do *hypervolume* HP(P*) gerado, em cada replicação *r*, pela fronte de Pareto P* da população presente em cada replicação *r* de uma iteração i (T_i em r), com T_r produzindo as populações finais de 1 a 10. Dito isto, nota-se que o *hypervolume* calculado para uma única replicação de otimização, HV(P(Ti)), converge em i=300 para o valor de 403.1808. Isso implica dizer que execuções além de 300 iterações provavelmente não contribuem para aumento de soluções efetivas na frente de Pareto final (P), pois o valor citado representa aproximadamente 95% do *hypervolume* relacionado à frente de Pareto de referência P*. Foi investigado também a hipótese nula H_o em cada uma das iterações para avaliar o fator de convergência do *hypervolume* na otimização de cada uma delas.

Foi analisado, em cada iteração i a hipótese nula de que a média do *hypervolume* é menor ou igual a 95% de seu valor final. O que pode ser traduzido para a expressão:

$$H_o : HV(P(T^i)) <= 0.95.HV(P(T^{300})) \quad (1)$$

onde o objetivo é encontrar o menor valor de iteração no qual esta hipótese é rejeitada, com base e um nível de significância α. Para tanto, foram aplicados os testes de Anderson-Darling, Levene, t-test e Wilcoxon R, conforme exibe a figura 13.

O teste de Anderson-Darling foi utilizado para verificar o aspecto de distributividade dos valores de *hypervolume* HV(P(T^i)) em cada iteração. O teste de Levene foi aplicado com o intuito de observar a homoscedasticidade[4] da hipótese entre os valores de *hypervolume* HV(P(T^i)) e HV(P(T^{300})). O valor 0.05 foi considerado como nível de significância α em ambos os testes.

As iterações de teste com p-value de Anderson-Darling $\geq$ 0.05 devem apresentar valores normalmente distribuídos e aquelas iterações de teste com *p-value* de Levene $\geq$ 0.05 devem apresentar homoscedasticidade nos valores de *hypervolume*. No caso de estas duas condições serem satisfeitas, a hipótese nula H_o deve então ser avaliada por métodos de teste paramétricos e do contrário, por métodos de teste não-paramétricos. Neste trabalho foram considerados: o método de teste paramétrico: t-test e o método de teste não-paramétrico Wilcoxon Signed-Rank (SR), ambos unicaudais, para conter essas condições.

Com um nível de significância α de 0.05, no cenário relatado, foi possível identificar que o *hypervolume* de uma única execução atinge 95% do seu valor final na iteração 223(primeira incidência de *p-value* com valor inferior a 0.05). Isso significa considerar que, a cada 100 execuções de otimização, 95 delas atingirá 95% do seu valor final na 215^a iteração.

Tomando como base a validação da correta captura das soluções arquiteturais próprias aos sistemas auto-organizáveis, identificadas neste trabalho, foi possível considerar que este trabalho contribui para a diminuição do tempo e complexidade de execução da atividade de projeto arquitetural de sistemas auto-organizáveis e do aumento de suporte ao apoio de decisão também para a realização desta tarefa. Conclui-se então, que, o objetivo geral: automatizar o processo de projeto arquitetural de sistemas auto-organizáveis e os objetivos específicos: catalogar as soluções arquiteturais próprias ao projeto de sistemas auto-organizáveis; facilitar a realização da atividade de projeto arquitetural para sistemas deste tipo; e propiciar o reuso de suas soluções foi cumprido neste trabalho.

O processo de automatização arquitetural proposto neste trabalho foi realizado sobre a ferramenta DuSE-MT. O cálculo dos valores de *hypervolume* foi realizado na ferramenta R com apoio do pacote emoa[5]. Os testes Anderon-Darling, t-test e Wilcoxon SR foram executados na ferramenta Scilab, também com apoio do módulo CASCILIB[6].

V. Trabalhos Correlatos

O SO:DuSE é um espaço de projeto DuSE para o propósito dos sistemas auto-organizáveis. O espaço de projeto SA:DuSE [9] trata dos sistemas *self-adaptive*. No entanto, ele não suporta as características específicas ao processo auto-organizável. O SO:DuSE é direcionado aqueles sistemas que se adequam a um contexto sem a necessidade de operação de um ator central e conhecimento da estrutura interna do modelo do sistema [5].

Anteriormente ao trabalho [9], na literatura de auto-organização, foram apresentadas ferramentas próprias à automatização da tarefa de projeto arquitetural como estes [26] [27]. Nenhum deles apresenta uma abordagem que envolva a disponibilização da implementação dos padrões arquiteturais envolvidos, assim como possibilita o seu *redesign* a um novo formato, contendo dimensões arquiteturais de livre escolha do ator da ferramenta. Neste outro [28] é abordada a criação de um *framework* baseado em um novo método para a verificação experimental desses sistemas. Mas ele também não define uma cobertura sistemática para os padrões arquiteturais envolvidos e seu problema está relacionado a uma área mais ampla do que a desta proposta, que é da Engenharia Reversa.

O trabalho [1], de 1990, apresenta algumas características consideradas primordiais à formação de redes auto-organizáveis confiáveis. Utilizando como base as pesquisas do campo da Inteligência Artificial, o autor definiu e explicou, teoricamente, os aspectos biológicos de auto-organização que poderiam ser utilizados no meio computacional.

O trabalho [29], de 2003, propõe a criação de um *framework* para facilitar o projeto de sistemas que passaram por transformações em sua arquitetura. Assim como o SO:DuSE ela utiliza a abordagem de atribuição de padrões arquiteturais para conduzir a seleção. No entanto ela possui a restrição de uso somente a um projeto local, o Xfig. O artigo [30], de 2004, descreve explicitamente um catálogo contendo primitivas inspiradas biologicamente para a engenharia de sistemas auto-organizáveis.

No artigo [31], de 2006, acontece a primeira tentativa atual de catalogar as principais funções encontradas em formações

[4]Homoscedasticidade = apresenta-se como uma forte dispersão dos dados em torno de uma reta; distribuição de frequência em que todas as distribuições condicionadas têm desvios padrão diferentes.

[5]http://cran.rstudio.com/ e https://cran.r-project.org/web/packages/emoa/

[6]http://www.scilab.org/ e https://atoms.scilab.org/toolboxes/casci

Iteração i	$\overline{HV}(P(T^i))$	*p-value* de Anderson-Darling	*p-value* de Levene	Teste Utilizado	*p-value* (*t-test* / *Wilcoxon SR*)
...					
222	12.200411	0.113852	0.274383	*t-test*	0.101627
223	23.574236	0.152420	0.260664	*t-test*	**0.045947**
224	23.574236	0.152420	0.260664	*t-test*	0.045947
225	24.272402	0.047992	0.320262	*Wilcoxon SR*	**0.009347**
226	24.272402	0.047992	0.320262	*Wilcoxon SR*	0.009347
...					

Figura 13. Resultados dos testes de hipótese para verificação de convergência do *hypervolume* na otimização

biológicas de auto-organização para padrões arquiteturais computacionais. Ele modela e disponibiliza os meios para projetar os sistemas computacionais auto-organizáveis. Junto com o trabalho [15], de 2007, eles formam a base para diversos trabalhos do campo. O autor deste último trabalho também agrupa alguns padrões arquiteturais voltados aos sistemas auto-organizáveis para implementação posterior.

O trabalhos [32] e [33], do ano de 2009 exibem, respectivamente, uma abordagem e uma linguagem própria à implementação dos padrões arquiteturais formulados para a construção de sistemas auto-organizáveis. Neste mesmo ano [34] apresenta um novo trabalho contendo a abordagem de padrões arquiteturais para sistemas auto-organizáveis, mas se refere somente àqueles baseados em informações químicas digitais. Ainda no ano de 2009 [35] apresenta a proposta de um *framework auto-organizável* próprio à extração de informações em *data centers*. O SOFIE comprova a aplicação desses tipos de sistemas em vários campos computacionais.

No ano de 2010, a tese [8] apresenta um guia sistemático para a implementação de diversos padrões arquiteturais no meio computacional. JASOF é na verdade uma biblioteca não disponibilizada livremente para a implementação de sistemas auto-organizáveis. Ela utiliza como base o *framework* Jadex e por isso oferece suporte às linguagens XML e Java. Apesar de oferecer uma proposta semelhante ao SO:DuSE, ele não possibilita o *redesign* de um projeto arquitetural para um segundo modelo, no formato de auto-organização. Além disso, ele está restrito à implementação de somente daqueles padrões arquiteturais definidos pelo autor como tipos básicos, ao passo que o SO:DuSE oferece uma gama maior de padrões arquiteturais. O JASOF implementa os seguintes padrões arquiteturais: *Replication, Evaporation, Aggregation* e *Diffusion*. Além disso, sua abstração visa servir de suporte à implementação de código-fonte, enquanto o SO:DuSE envolve-se a nível de implantação também.

Por fim, o trabalho [3], um catálogo de padrões arquiteturais auto-organizáveis do ano de 2012, foi utilizado como base para a escolha dos padrões arquiteturais disponibilizados no SO:DuSE. Isso porque ele fundamenta e apresenta as funções que estes padrões arquiteturais exercem sobre os auto-organizáveis, possibilitando o entendimento e dos mecanismos envolvidos nos processos relacionados à elas. O trabalho [18] foi utilizado como base para a construção dos modelos arquiteturais dispostos no SO:DuSE e o exemplo de estudo de caso do almoxarifado, porque ele contém definições de arquitetura específicas aos sistemas auto-organizáveis.

Todos estes trabalhos apresentam uma abordagem voltada ao assunto de auto-organização aplicada aos sistemas computacionais. No entanto, ainda que definam arquiteturas de referência, métodos e *frameworks* para este fim, eles não disponibilizam os padrões arquiteturais auto-organizáveis já consolidados na literatura de maneira a tornar possível a refatoração de modelos de sistemas genéricos em sistemas descentralizados auto-organizáveis assim como a proposta SO:DuSE objetiva fazer.

VI. Conclusão e Trabalhos Futuros

O aumento da complexidade de *software* vem sendo considerado como um dos fatores limitantes à construção e manutenção dos novos sistemas computacionais. Projetar novos sistemas conforme característica de auto-organização tem sido apontado como um meio de contornar esta condição.

Existe uma variedade de técnicas propícias à implementação de auto-organização sobre os sistemas computacionais que utilizam esse tipo de coordenação. No entanto, a maioria dos métodos disponíveis pertencem à Engenharia de Software convencional, sendo necessário ajustá-los, combiná-los ou implementar novos métodos propícios aos sistemas deste tipo. A Engenharia de Software Baseada em Busca dispõe técnicas de otimização que possibilitam combinar soluções candidatas à resolução de um problema para extrair as melhores a dado(s) objetivo(s).

Os padrões arquiteturais são soluções amplamente utilizadas e testadas e por isso podem ser consideradas soluções confiáveis para a implementação de sistemas. A catalogação sistemática de soluções deste tipo representa importante contribuição ao meio e confere em grande auxílio na atividade de projeto arquitetural, o que pode influenciar diretamente na efetividade e qualidade dos modelos resultantes. Ainda, automatizar a atividade de modelagem arquitetural, sobretudo para sistemas auto-organizáveis, também constitui relevante evento ao meio.

O espaço de projeto SO:DuSE foi desenvolvido com o intuito de permitir a automatização de processo de projeto arquitetural. Ele foi criado especificamente para os sistemas

auto-organizáveis e segue o formato de processo e implementação do metamodelo arquitetural DuSE. Nele estão dispostos 4 dimensões de projeto e treze pontos de variação agrupados em soluções arquiteturais dos tipos *Spreading* ou *Gradient*.

A ferramenta foi validada com o intuito de certificar a correta captura dos modelos arquiteturais projetados neste trabalho e com isso facilitar a atividade de projeto arquitetural para sistemas auto-organizáveis e garantir o reuso dessas soluções. Sobre a ferramenta DuSE-MT essas soluções puderam ser combinadas para extrair soluções viáveis, contendo: alta escalabilidade, baixa sobrecarga de comunicação e baixa complexidade arquitetural. No total, foram identificadas 3 arquiteturais que compunham este formato de projeto. Essas soluções, assim como todas as execuções de otimização realizadas para este fim foram avaliadas conforme comparação de resultados em uma matriz de gráficos de dispersão de soluções, definição de hipótese nula e aplicação de testes para *hypervolume*, parametrizado e não parametrizado para esta confirmação.

Apesar de a ferramenta apresentar resultado satisfatório, faz-se necessário ampliar seu espaço de busca por identificar e modelar novas dimensões e pontos de variação ao espaço de projeto SO:DuSE. Esta ação viabiliza aplicar com menos restrições a otimização e assim exercitar mais seu espaço de busca para obter resultados de projeto mais proveitosos aos sistemas auto-organizáveis. Ele também pode ser adaptado para aderir à outras características próprias destes sistemas, como a captura de configuração de tempo, para uso dos sistemas provenientes destes modelos em *runtime*.

Referências

[1] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, pp. 1464–1480, 1990.

[2] G. Muehl, M. Werner, M. A. Jaeger, K. Herrmann, and H. Parzyjegla, "On the definitions of self-managing and self-organizing systems," in *Communication in Distributed Systems (KiVS), 2007 ITG-GI Conference*, Feb 2007, pp. 1–11.

[3] J. L. Fernandez-Marquez, *Bio-inspired Mechanisms for Self-organising Systems*. CSIC Press, 2012.

[4] G. D. M. Serugendo, N. Foukia, S. Hassas, A. Karageorgos, S. K. Mostéfaoui, O. F. Rana, M. Ulieru, P. Valckenaers, and C. van Aart, "Self-organisation: Paradigms and applications." in *Engineering Self-Organising Systems*, ser. Lecture Notes in Computer Science, vol. 2977. Springer, 2003, pp. 1–19. [Online]. Available: http://dblp.uni-trier.de/db/conf/atal/esoa2003.html#SerugendoFHKMRUVA03

[5] H. V. D. Parunak and S. A. Brueckner, "Software engineering for self-organizing systems," in *12th International Workshop on Agent-Oriented Software Engineering (AOSE 2011)*, D. Weyns and J. P. Müller, Eds., AAMAS 2011, Taipei, Taiwan, 2 May 2011.

[6] W. Banzhaf, "Self-organizing systems," in *Encyclopedia of Complexity and Systems Science*, 2009, pp. 8040–8050.

[7] Z. Zhao, C. Gao, and F. Duan, "A survey on autonomic computing research," in *Computational Intelligence and Industrial Applications, 2009. PACIIA 2009. Asia-Pacific Conference on*, vol. 2, Nov 2009, pp. 288–291.

[8] M. T. de Abreu Netto, "Um framework baseado em padrões para a construção de sistemas multi-agentes auto-organizáveis," Master's thesis, Pontífica Universidade Católica do Rio de Janeiro, 2010. [Online]. Available: http://www.maxwell.lambda.ele.puc-rio.br/Busca_etds.php?strSecao=resultado&nrSeq=16434@1

[9] S. S. Andrade and R. J. de A. Macedo, "A search-based approach for architectural design of feedback control concerns in self-adaptive systems," in *Self-Adaptive and Self-Organizing Systems (SASO), 2013 IEEE 7th International Conference on*, 2013, pp. 61–70.

[10] Y. Guo, X. Mao, C. Hu, J. Yin, and J. Cao, "A survey of software engineering for self-organization systems." in *SEKE*. Knowledge Systems Institute Graduate School, 2011, pp. 539–542. [Online]. Available: http://dblp.uni-trier.de/db/conf/seke/seke2011.html#GuoMHYC11

[11] M. Harman, P. McMinn, J. T. de Souza, and S. Yoo, "Empirical software engineering and verification," B. Meyer and M. Nordio, Eds. Berlin, Heidelberg: Springer-Verlag, 2012, ch. Search Based Software Engineering: Techniques, Taxonomy, Tutorial, pp. 1–59. [Online]. Available: http://dl.acm.org/citation.cfm?id=2184075.2184076

[12] O. Räihä, "Survey: A survey on search-based software design," *Comput. Sci. Rev.*, vol. 4, no. 4, pp. 203–249, Nov. 2010. [Online]. Available: http://dx.doi.org/10.1016/j.cosrev.2010.06.001

[13] S. Andrade, "Projeto arquitetural automatizado de sistemas self-adaptive," Ph.D. dissertation, UFBA, 2014.

[14] S. S. Andrade and R. J. de A. Macedo, *DuSE-MT: From Design Spaces to Automated Software Architecture Design (SEKE 2013 Best Demo Award)*, 2013.

[15] L. Gardelli, M. Viroli, and A. Omicini, "Design patterns for self-organising systems," in *Multi-Agent Systems and Applications V*, ser. Lecture Notes in Computer Science, H.-D. Burkhard, G. Lindemann, R. Verbrugge, and L. Varga, Eds. Springer Berlin Heidelberg, 2007, vol. 4696, pp. 123–132. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-75254-7_13

[16] T. D. Wolf and T. Holvoet, "Design patterns for decentralised coordination in self-organising emergent systems." in *ESOA*, ser. Lecture Notes in Computer Science, vol. 4335. Springer, 2007, pp. 28–49. [Online]. Available: http://dblp.uni-trier.de/db/conf/esoa/esoa2006.html#WolfH06

[17] A. Pitangueira, R. Maciel, M. de Oliveira Barros, and A. Andrade, "A systematic review of software requirements selection and prioritization using sbse approaches," in *Search Based Software Engineering*, ser. Lecture Notes in Computer Science, G. Ruhe and Y. Zhang, Eds. Springer Berlin Heidelberg, 2013, vol. 8084, pp. 188–208. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-39742-4_15

[18] J. Juziuk, D. Weyns, and T. Holvoet, "Design patterns for multi-agent systems: A systematic literature review," in *Agent-Oriented Software Engineering*, O. Shehory and A. Sturm, Eds. Springer Berlin Heidelberg, 2014, pp. 79–99. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-54432-3_5

[19] M. Rubenstein, C. Ahler, and R. Nagpal, "Kilobot: A low cost scalable robot system for collective behaviors," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, May 2012, pp. 3293–3298.

[20] M. Rubenstein, A. Cabrera, J. Werfel, G. Habibi, J. McLurkin, and R. Nagpal, "Collective transport of complex objects by simple robots: Theory and experiments," in *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, ser. AAMAS '13. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2013, pp. 47–54. [Online]. Available: http://dl.acm.org/citation.cfm?id=2484920.2484932

[21] T. Wijayasiriwardhane and R. Lai, "Component point: A system-level size measure for component-based software systems," *Journal of Systems and Software*, vol. 83, no. 12, pp. 2456 – 2470, 2010.

[22] M. Mamei and F. Zambonelli, "Motion coordination in the quake 3 arena environment: A field-based approach," in *Environments for Multi-Agent Systems*, ser. Lecture Notes in Computer Science, D. Weyns, H. Van Dyke Parunak, and F. Michel, Eds. Springer Berlin Heidelberg, 2005, vol. 3374, pp. 264–278. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-32259-7_14

[23] ——, "Theory and practice of field-based motion coordination in multiagent systems," *J. Appl. Artif. Intell*, vol. 19, 2005.

[24] M. Mamei, L. Leonardi, M. Mahan, and F. Zambonelli, "Motion coordination for ubiquitous agents."

[25] M. Mamei, R. Menezes, R. Tolksdorf, and F. Zambonelli, "Case studies for self-organization in computer science," *Journal of Systems Architecture*, vol. 52, no. 8-9, pp. 443–460, 2006.

[26] K. Sartipi, "Alborz: a query-based tool for software architecture recovery," in *Program Comprehension, 2001. IWPC 2001. Proceedings. 9th International Workshop on*, 2001, pp. 115–116.

[27] D. R. Wallace, "A design tool architecture for the rapid evaluation of product design tradeoffs in an inernet-based system modeling environment." [Online]. Available: http://hdl.handle.net/1721.1/32375

[28] B. de Castro Bahia Alvarenga Soares, "Um método e um framework para o planejamento empírico de sistemas multiagentes auto-organizáveis," Master's thesis, Pontífica Universidade Católica do Rio de Janeiro, 2010.
[29] K. Sartipi, "Pattern-based software architecture recovery," in *In Proc. of the Second ASERC Workshop on Software Architecture*, 2003, p. 2003.
[30] R. Nagpal, "A catalog of biologically-inspired primitives for engineering self-organization," in *Engineering Self-Organising Systems*, ser. Lecture Notes in Computer Science, G. Di Marzo Serugendo, A. Karageorgos, O. Rana, and F. Zambonelli, Eds. Springer Berlin Heidelberg, 2004, vol. 2977, pp. 53–62. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-24701-2_4
[31] T. D. Wolf and T. Holvoet, "A catalogue of decentralised coordination mechanisms for designing self-organising emergent applications," in *CW 458, DEP. OF COMPUTER SCIENCE*. Katholieke Universiteit Leuven, 2006, pp. 40–61.
[32] M. A. de Cerqueira Gatti, U. B. Sangiorgi, and C. J. P. de Lucena, "Towards a model driven approach for engineering self-organizing multi-agent systems," 2009.
[33] M. A. de Cerqueira Gatti, C. J. P. de Lucena, and A. F. Garcia, "A pattern language for self-organizing systems," 2009.
[34] H. Kasinger, B. Bauer, and J. Denzinger, "Design pattern for self-organizing emergent systems based on digital infochemicals," *2010 Seventh IEEE International Conference and Workshops on Engineering of Autonomic and Autonomous Systems*, vol. 0, pp. 45–55, 2009.
[35] F. M. Suchanek, M. Sozio, and G. Weikum, "Sofie: A self-organizing framework for information extraction," in *Proceedings of the 18th International Conference on World Wide Web*, ser. WWW '09. New York, NY, USA: ACM, 2009, pp. 631–640. [Online]. Available: http://doi.acm.org/10.1145/1526709.1526794

VII. APPENDIX

[7]

Dimensão de Projeto Infraestrutura de Comunicação

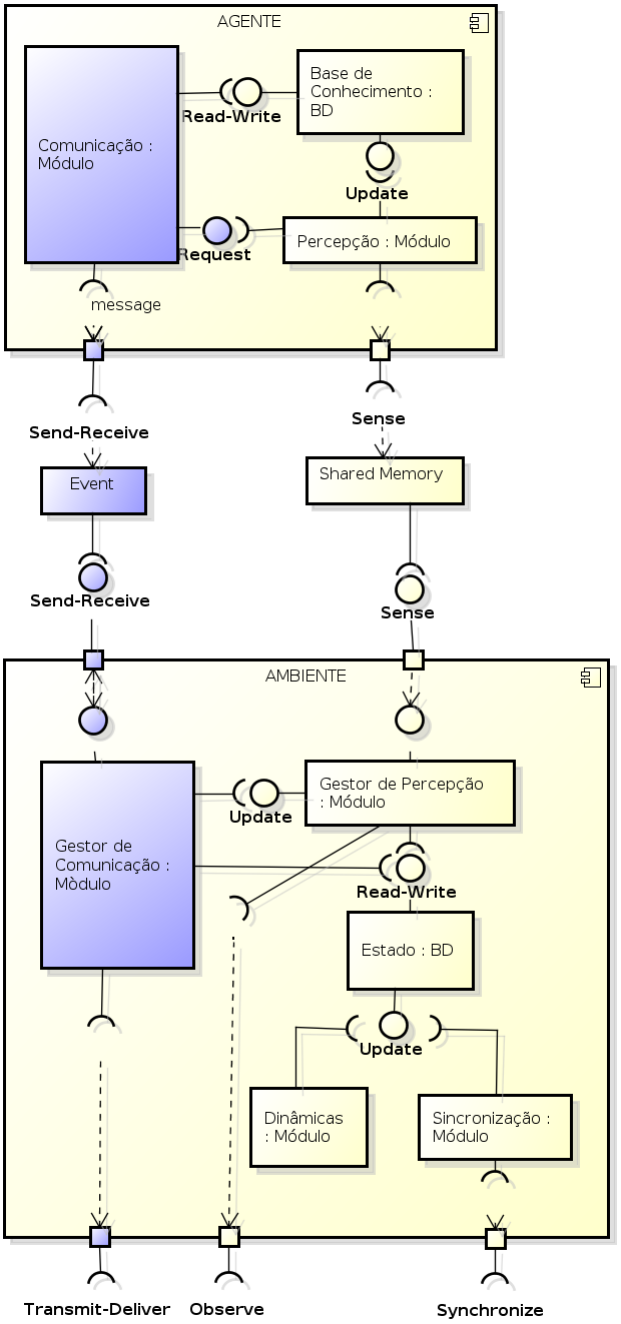

Figura 14. Ponto de Variação Troca de Mensagens

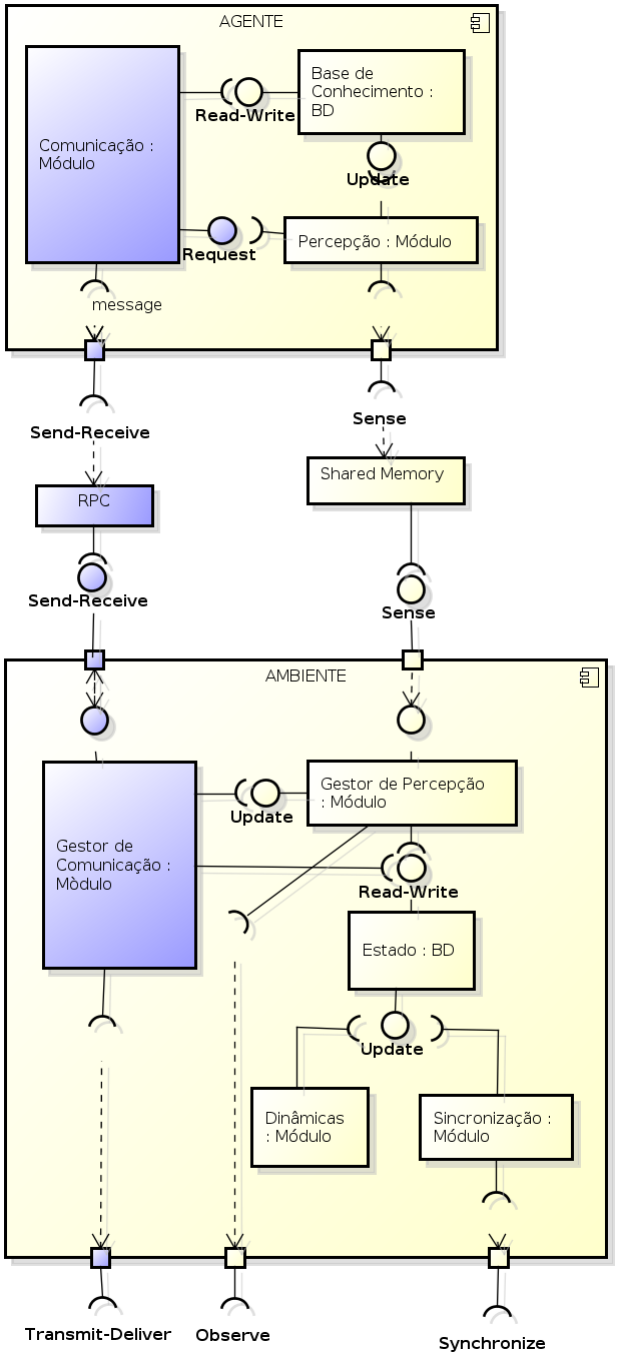

Figura 15. Ponto de Variação Chamada Remota

[7]**Legenda**:
Cor amarela: componentes da arquitetura inicial
Cor lilás e azul: componentes da nova arquitetura
Cor laranja: componentes utilizados de outra arquitetura <spreading ou field [gradient ou digital pheromone] >

Dimensão de Projeto Técnica de Comunicação

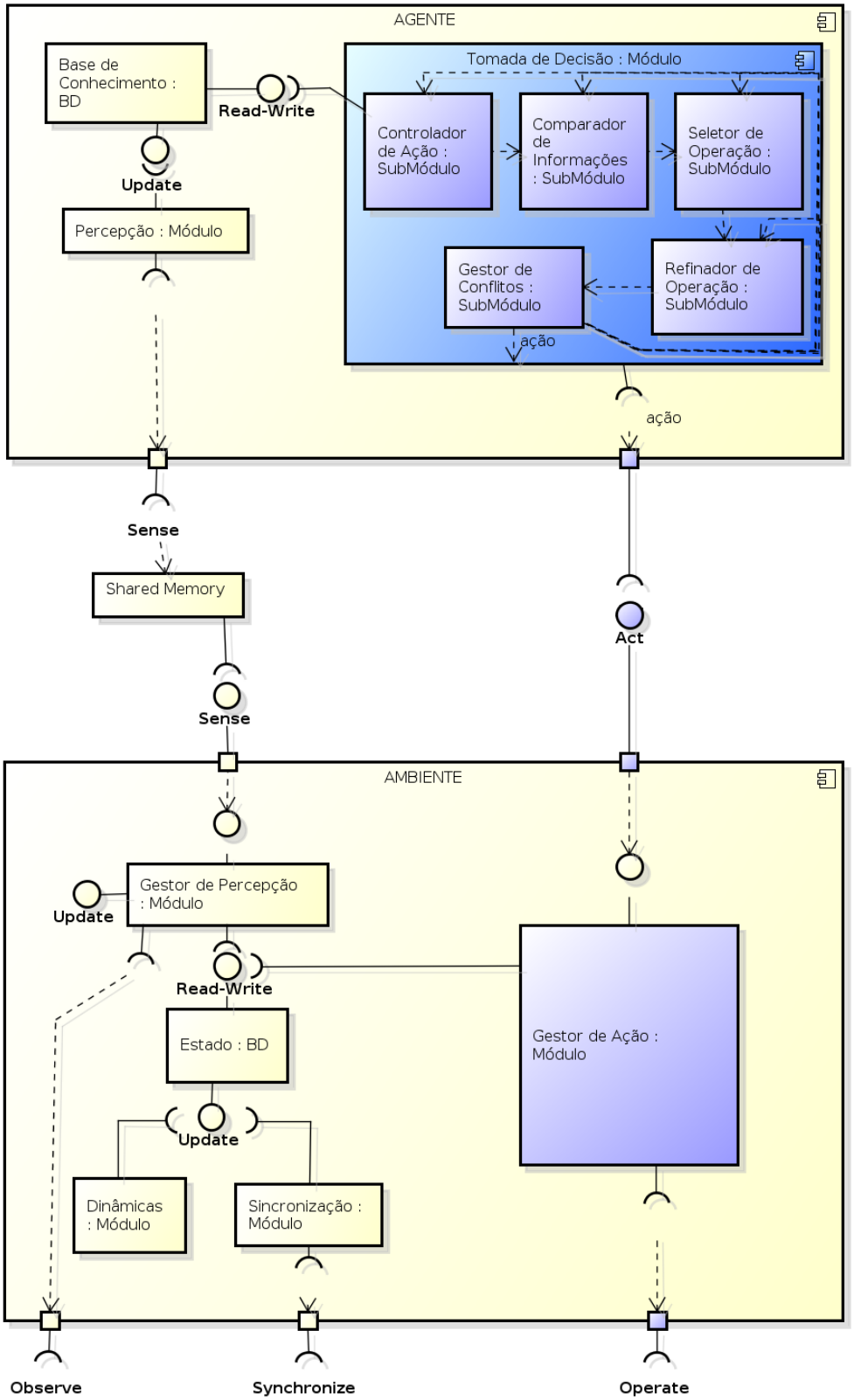

Figura 16. Ponto de Variação *Spreading*

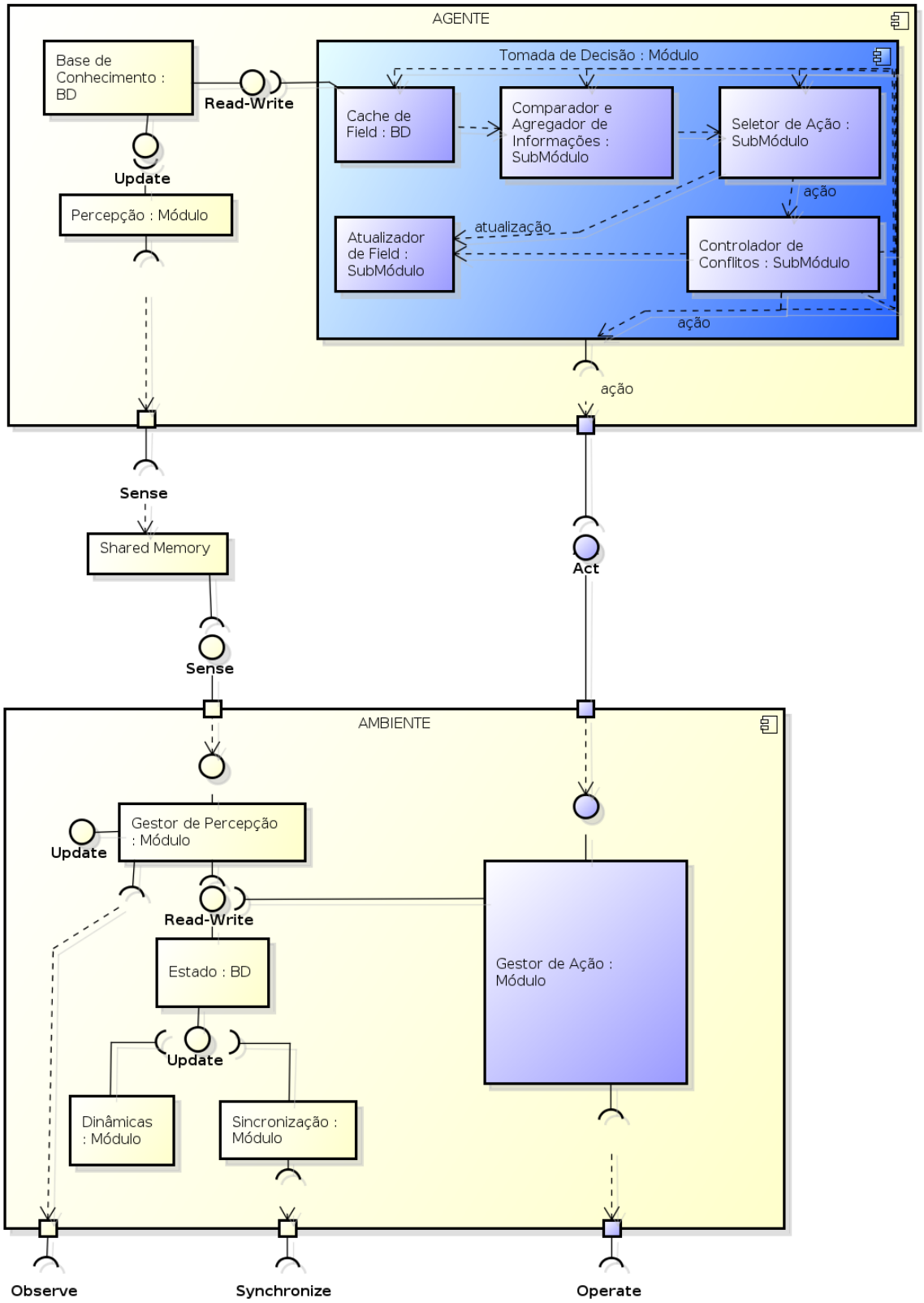

Figura 17. Ponto de Variação *Gradient*

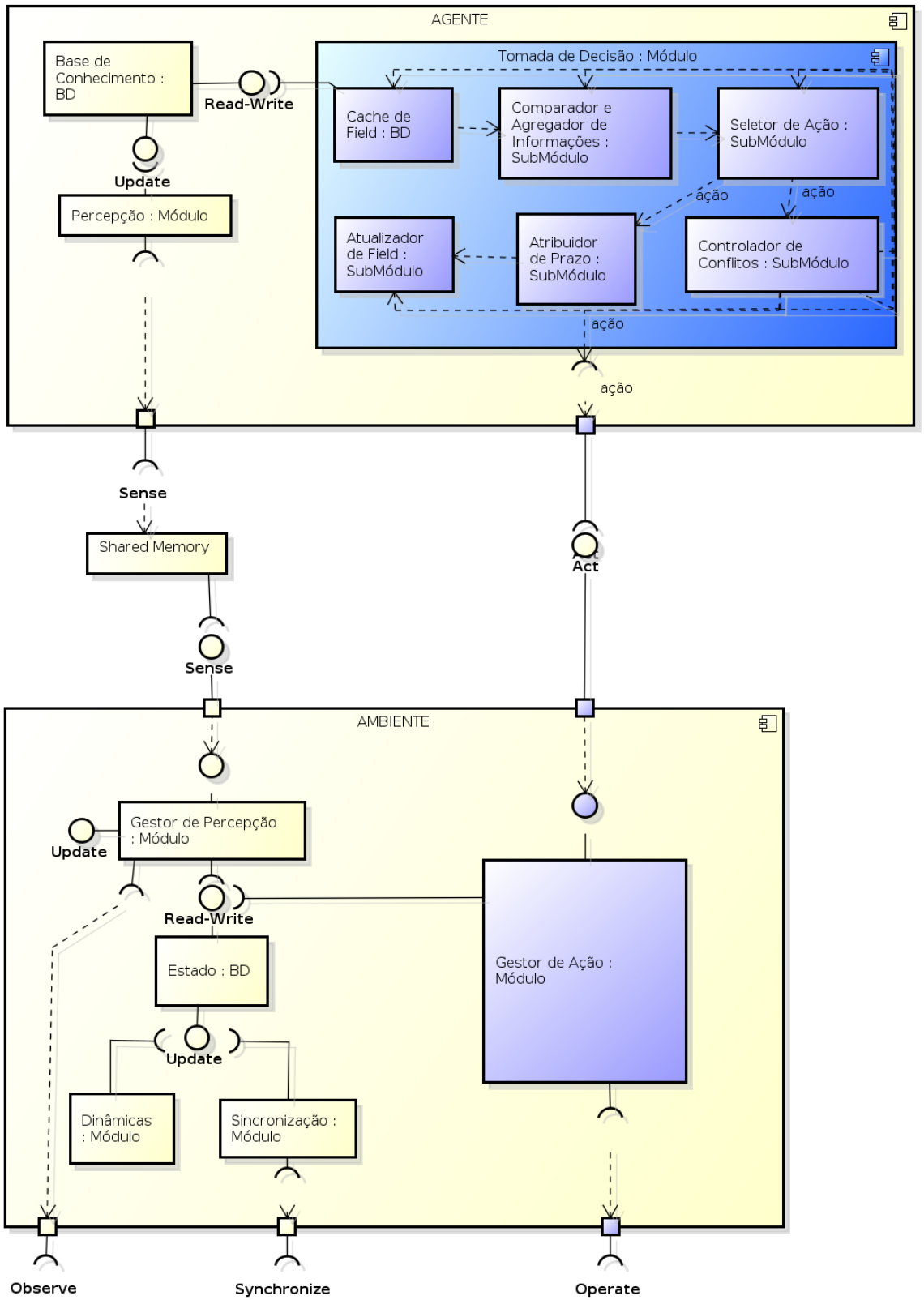

Figura 18. Ponto de Variação *Digital Pheromone*

Dimensão de Projeto Técnica de Coordenação de Movimentos

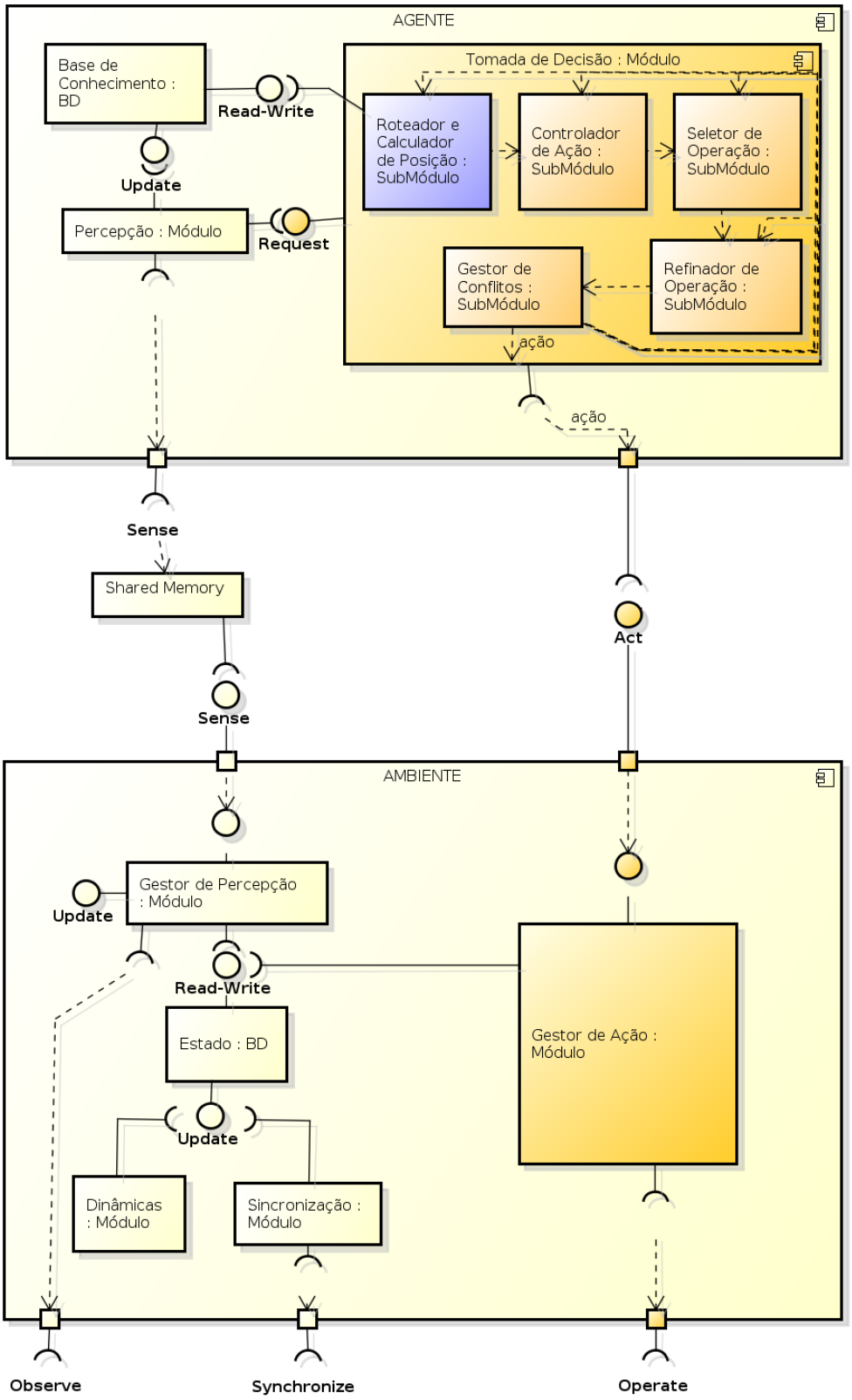

Figura 19. Ponto de Variação *Repulsion* baseado em *Spreading*

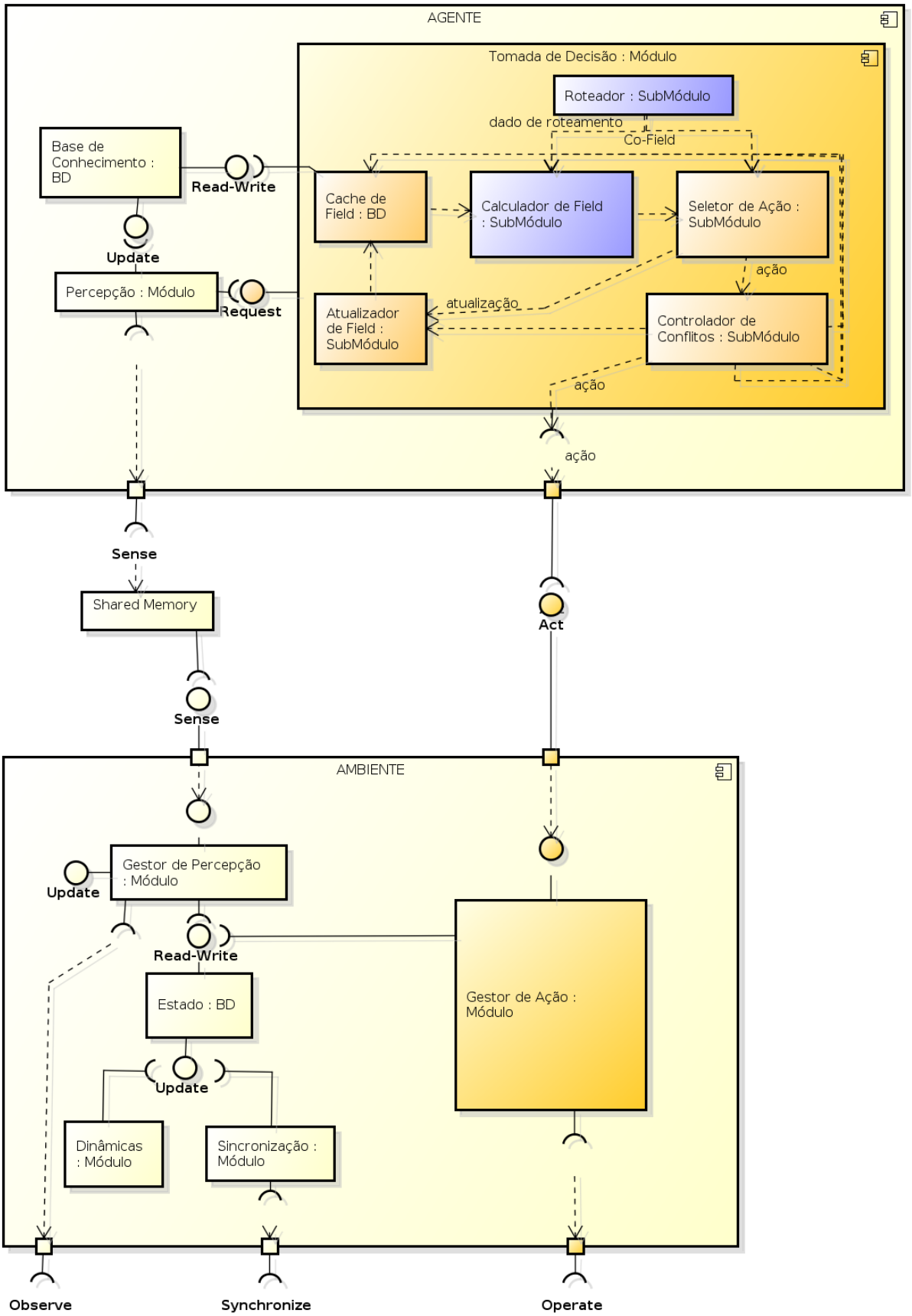

Figura 20. Ponto de Variação *Repulsion* baseado em *Field*

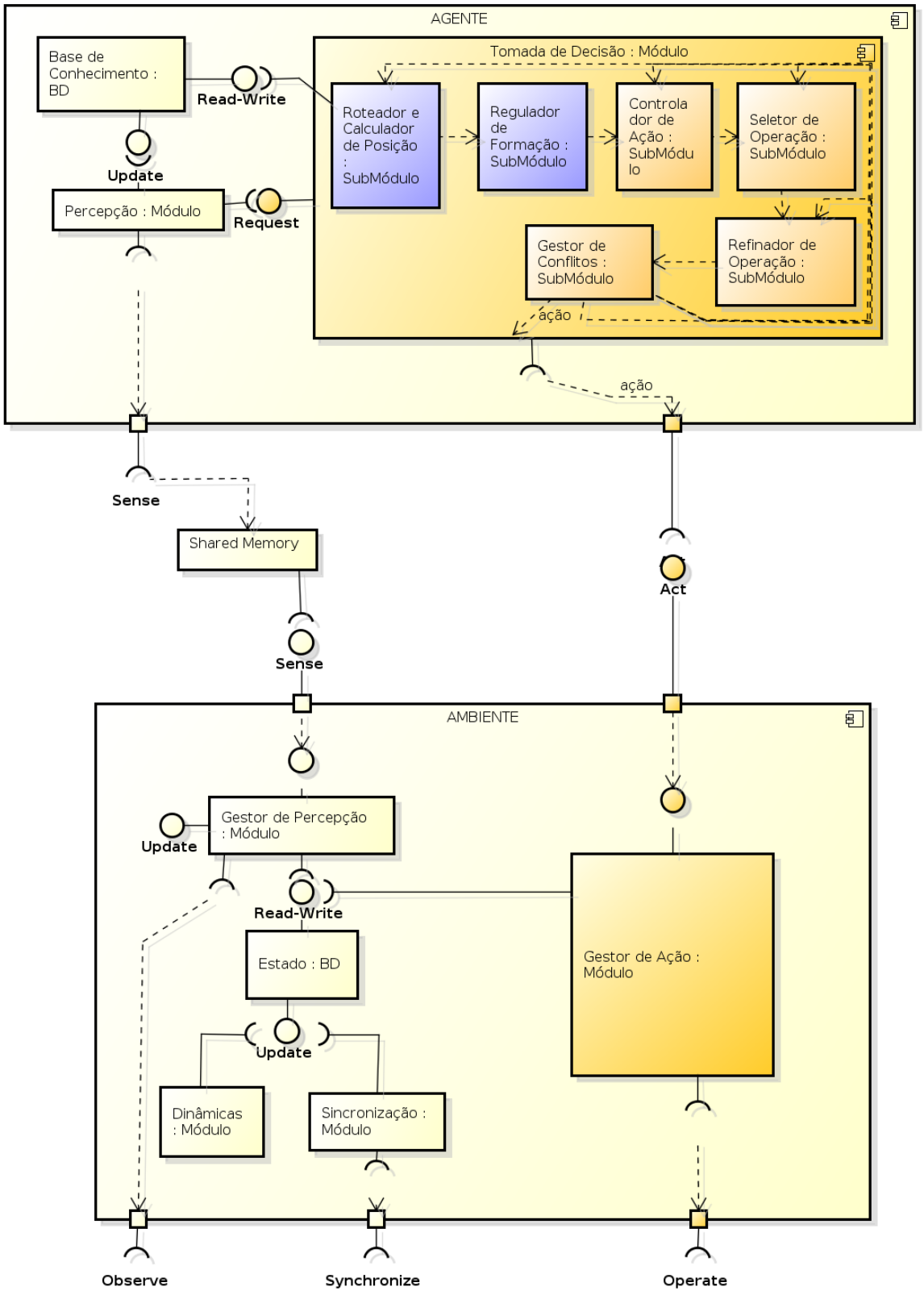

Figura 21. Ponto de Variação *Flocking* baseado em *Spreading*

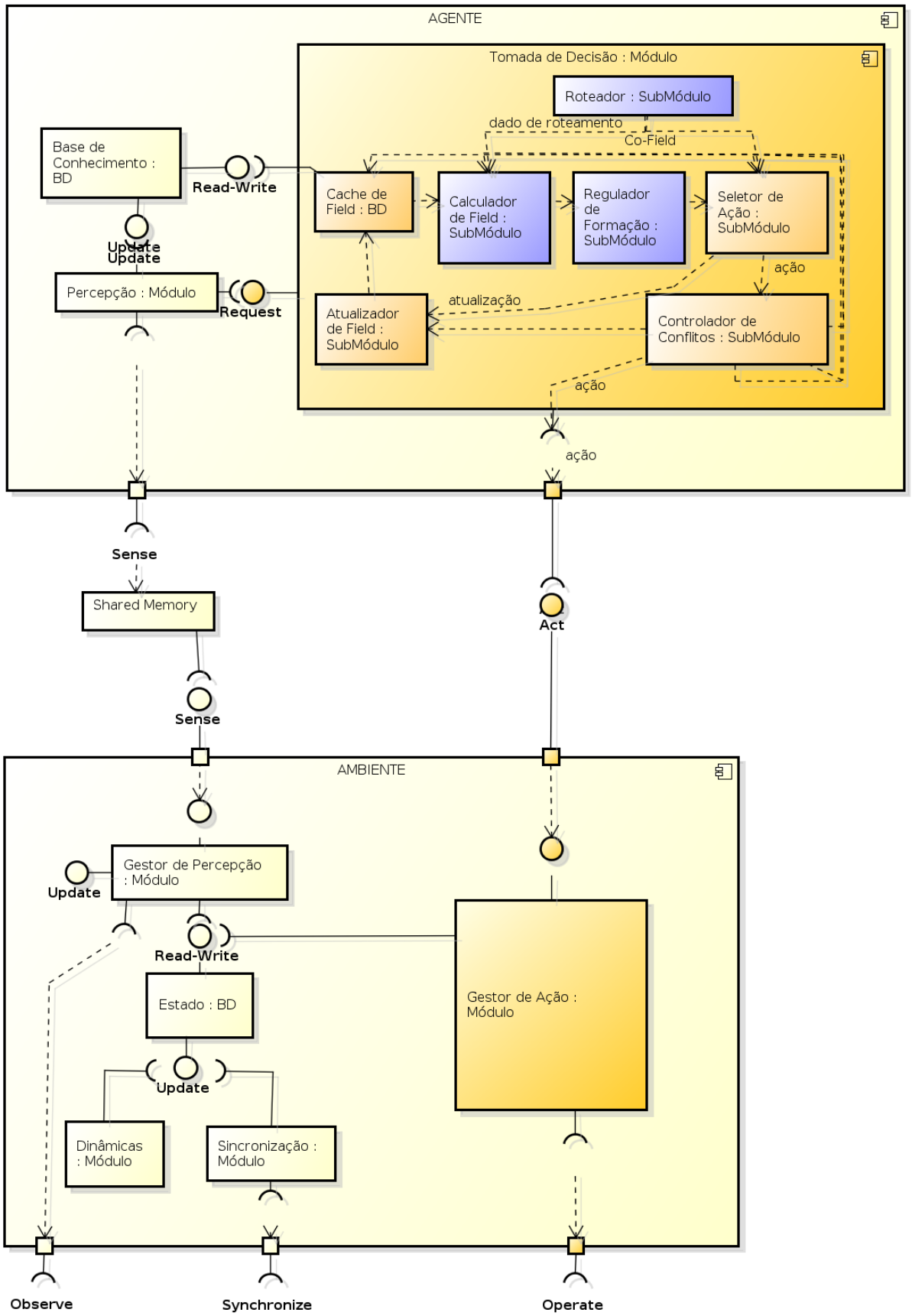

Figura 22. Ponto de Variação *Flocking* baseado em *Field*

Dimensão de Projeto Técnica de Tomada de Decisão

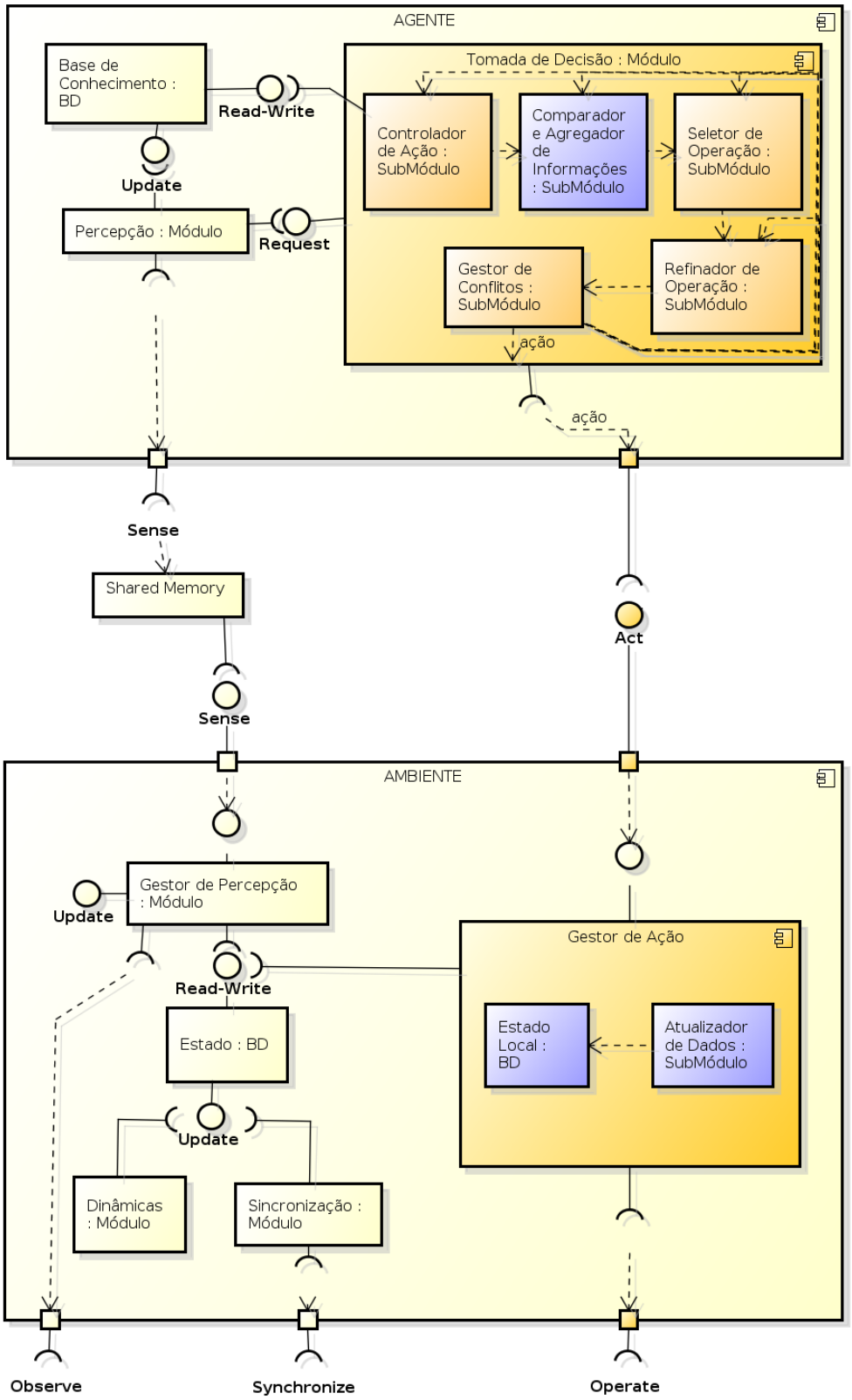

Figura 23. Ponto de Variação *Gossip* baseado em *Spreading*

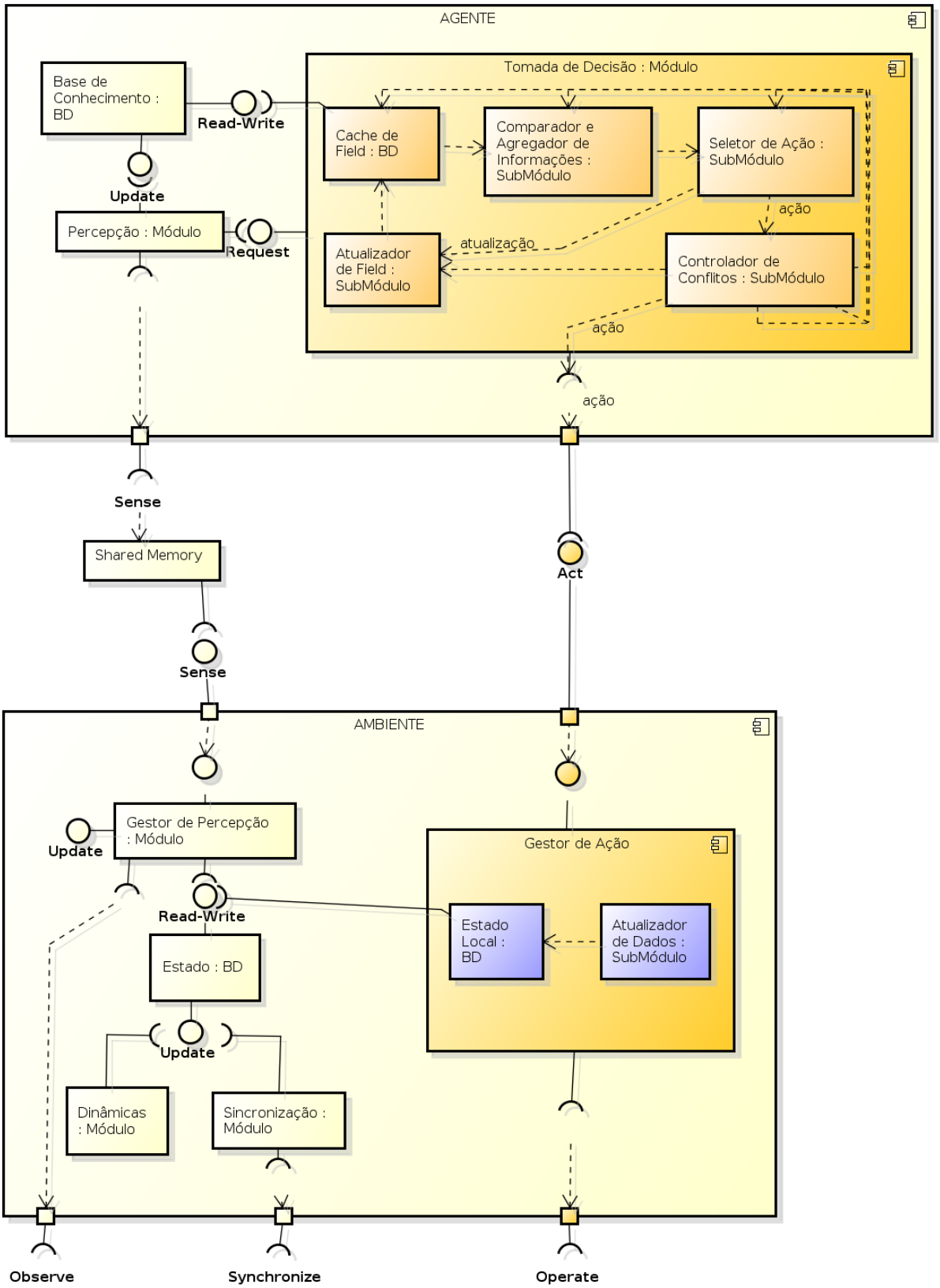

Figura 24. Ponto de Variação *Gossip* baseado em *Field*

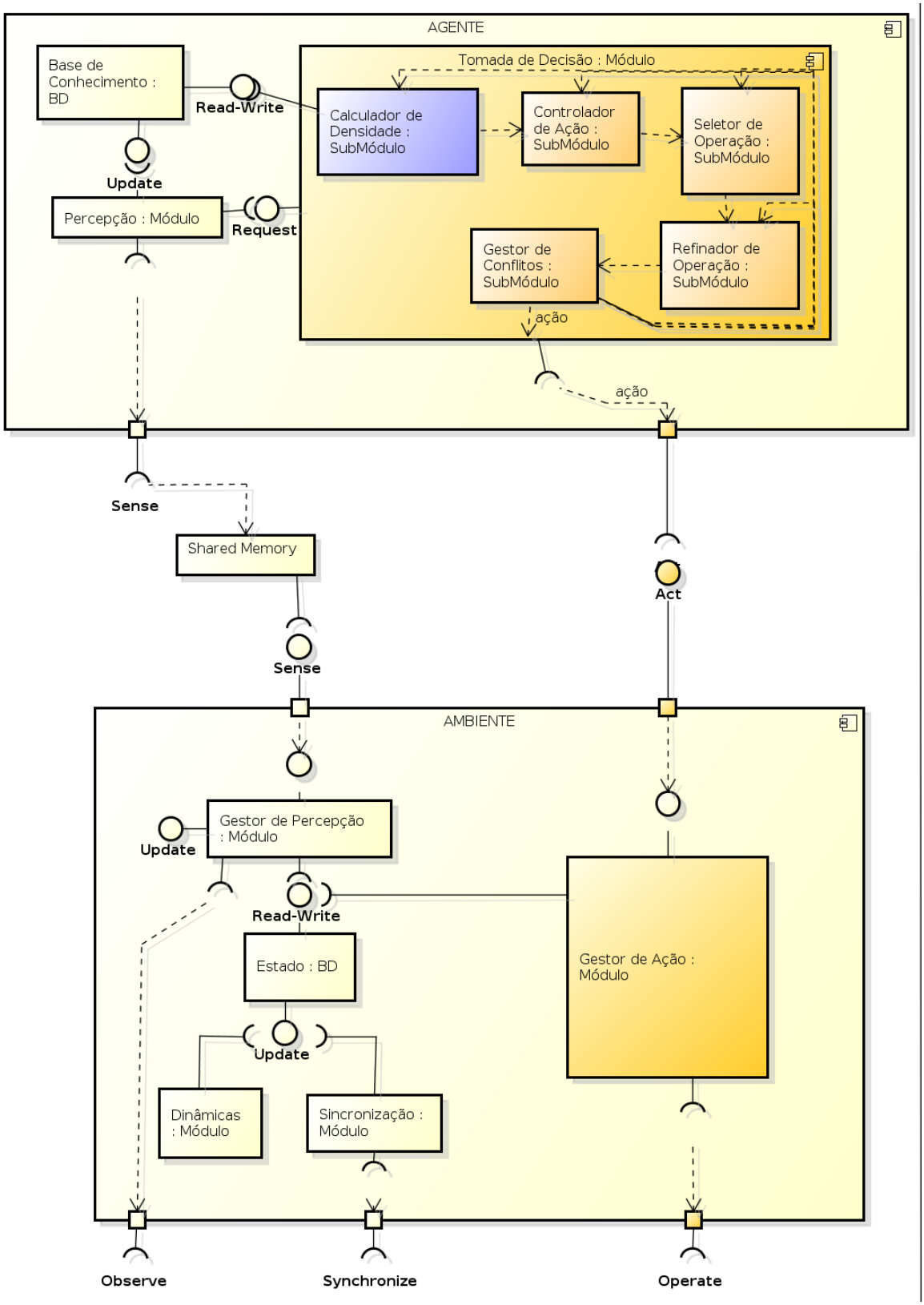

Figura 25. Ponto de Variação *Quorum Sensing* baseado em *Spreading*

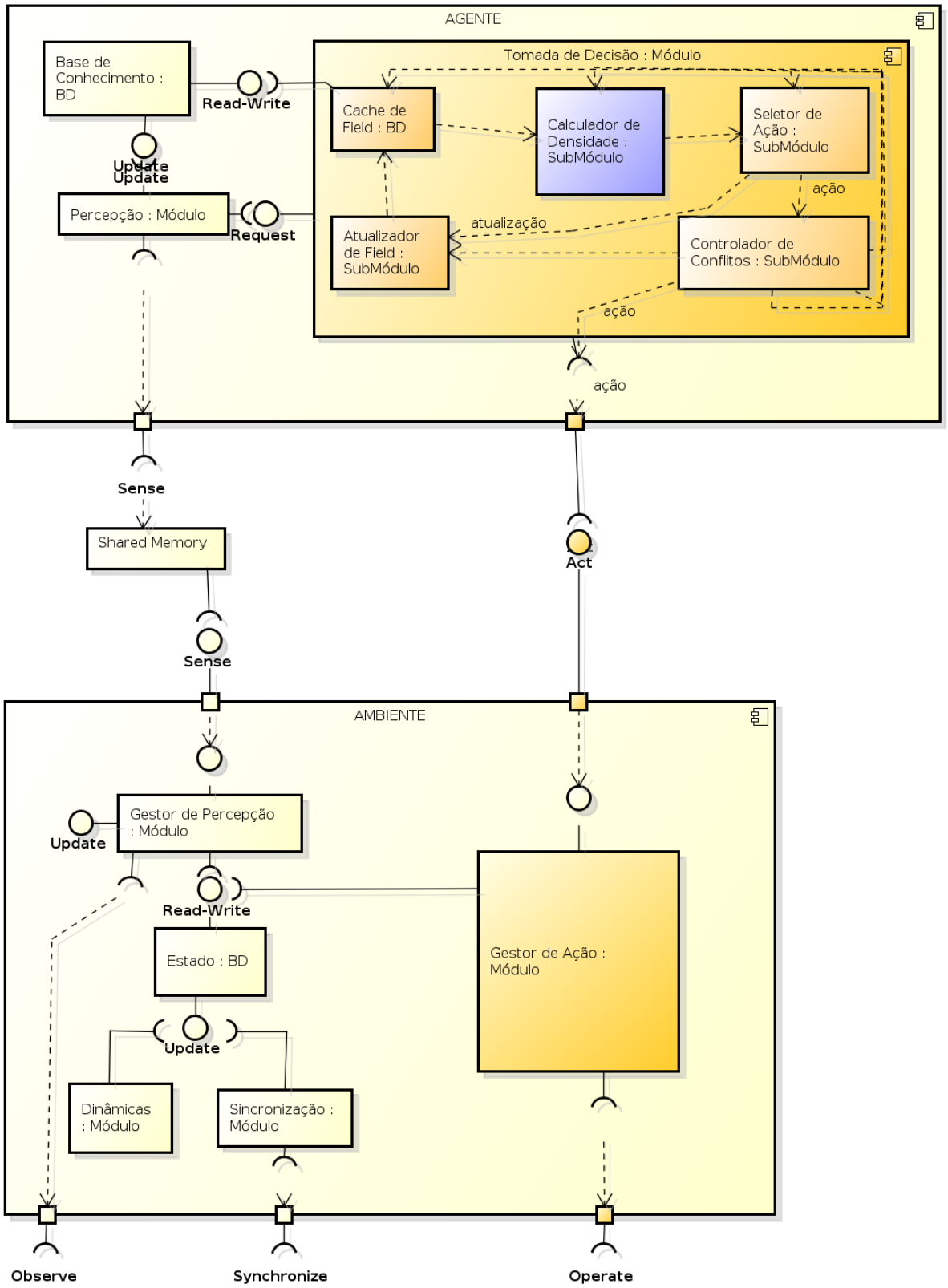

Figura 26. Ponto de Variação *Quorum Sensing* baseado em *Field*