# 3
# Tratamento de Exceções

Segundo a terminologia proposta por Anderson e Lee (1990), diz-se que um componente ou sistema tem um *defeito* se ele deixa de oferecer as funcionalidades previstas ou o faz em desacordo com sua especificação. Portanto, defeito é algo observável externamente pelo usuário do sistema. Diz-se também, que existe um *erro* no componente ou sistema quando existe um estado interno que sob determinadas entradas é propenso à ocorrência de um defeito. Além disso, podem existir erros que não provocam defeitos no sistema.

Uma *falha* é qualquer evento, ou seqüência de eventos, que propicia o surgimento de um erro. A existência de falhas é, portanto, inerente aos sistemas. Uma falha provoca um erro no estado do sistema, que pode se propagar até afetar o seu comportamento, resultando no aparecimento de um defeito (Lee e Anderson, 1990). *Tolerância a falhas* pode ser então definida como a capacidade de um sistema computacional de preservar suas funcionalidades, sem apresentar defeitos, mesmo na presença de falhas (Lee e Anderson, 1990). Desta forma, em situações de erro, um componente gera exceções que modelam a condição de erro e o sistema tolerante a falhas deve realizar o tratamento daquelas exceções.

Quando um componente é capaz de atender satisfatoriamente a uma requisição de serviço, se necessário, invocando serviços de outros componentes, ele retorna uma "resposta normal". Por outro lado, quando o serviço não pode ser realizado, ele retorna uma "resposta anormal", ou seja uma exceção. Exceções podem ser classificadas em três categorias: (i) uma *exceção de interface* é sinalizada em resposta a uma requisição de um serviço não disponível ou um serviço invocado com um conjunto inválido de entradas; (ii) uma *exceção interna* é levantada quando um componente detecta uma situação inesperada durante sua atividade normal e um tratador local é ativado; e (iii) uma *exceção de defeito* é sinalizada, indicando que, por algum motivo, um componente não pode prover o serviço especificado.

Quando um componente recebe uma resposta anormal de um outro componente (exceção de interface ou exceção de defeito) ou levanta uma exceção durante sua atividade normal (exceção interna), ele deve invocar os mecanismos apropriados de tolerância a falhas. Se estas exceções são tratadas, o componente pode voltar a prover o serviço normal. Entretanto, se o componente não consegue tratar a exceção, ele sinaliza a exceção para um componente de mais alto nível. A cada nível do sistema, um componente, chamado *componente tolerante a falhas ideal* (Figura 5), tratará as exceções produzidas pelos componentes de nível mais baixo ou irá propagá-las.

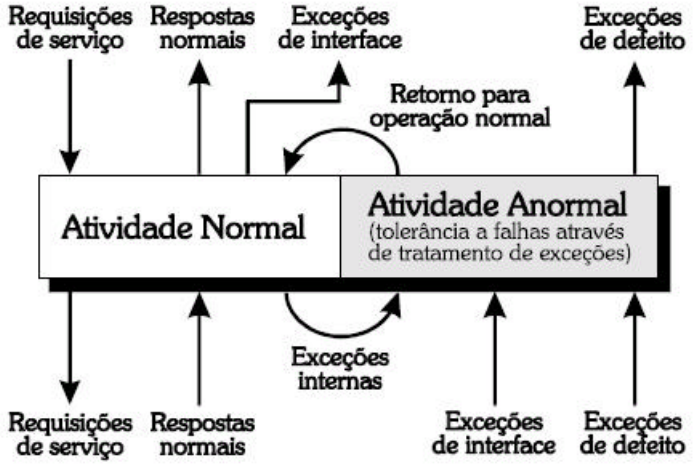


**Figura 5.** Componente Tolerante a Falhas Ideal (Lee & Anderson, 1990)

Adicionalmente, de acordo com Garcia et al. (2001), um tratamento de exceções adequado é responsável por: (i) identificar a causa de um erro, ou seja, a falha que provocou o estado errôneo; (ii) identificar que componente do sistema falhou; (iii) isolar o componente que falhou para que não afete os outros componentes do sistema; (iv) determinar a extensão do erro ocorrido; (v) recuperar o estado interno do sistema; (vi) reconfigurar o sistema para retomar o processamento normal e (vii) reiniciar o processamento normal. Em outras palavras, para Garcia et al. (2001), tratamento de exceções é a capacidade que um software possui de reagir apropriadamente diante da ocorrência de exceções, continuando ou interrompendo sua execução, a fim de preservar a máxima integridade possível no estado do sistema.

Um *tratador de exceções* em um programa constitui a parte da atividade excepcional, ou seja, a parte do código da aplicação que fornece medidas específicas da aplicação para o tratamento da exceção levantada (Garcia et al,

2001). Um tratador é vinculado a uma região particular do código normal, chamada *região protegida* ou *escopo de tratamento*. Se uma exceção é levantada em uma região protegida, o fluxo de controle normal é desviado para um fluxo de controle excepcional (Garcia et al., 2001).

Desenvolvedores de sistemas confiáveis freqüentemente se referem a erros como exceções porque erros raramente se manifestam durante a atividade normal do sistema. Exceções podem ser classificadas em dois tipos (Goodenough, 1975): (i) definidas pelo usuário, e (ii) pré-definidas. As exceções definidas pelo usuário são definidas e detectadas no nível de aplicação. As exceções pré-definidas são declaradas implicitamente e estão associadas com as condições errôneas detectadas pela máquina virtual, *middleware* ou *hardware* em tempo de execução.

Um mecanismo de tratamento de exceções introduz uma forma específica de propagação e mudanças no fluxo de controle normal para o fluxo de controle excepcional, quando uma exceção é levantada. Ele também é responsável por suportar diferentes estratégias de fluxo excepcional e procurar os tratadores apropriados depois que a ocorrência de uma exceção for detectada. Mecanismos de tratamento de exceções são construídos como uma parte inerente das linguagens de programação, com sua própria sintaxe ou são uma característica de *middlewares* que tratam com complexidades de diferentes domínios de aplicação e estilos de arquitetura.

O código relacionado à detecção e ao tratamento de exceções em sistemas confiáveis é freqüentemente numeroso e complexo, podendo ser até mesmo mais de dois terços de um programa (Garcia et al., 2001; Cristian, 1989; Gehani, 1992). Por este motivo é fundamental que os mecanismos de tratamento de exceções possuam um projeto simples, sejam fáceis de usar e também forneçam separação explícita entre o código normal e excepcional (Garcia et al., 2001; Iliasov & Romanovsky, 2005b).

Neste capítulo, inicialmente introduzimos os aspectos gerais da atividade de tratamento de exceções em sistemas. Posteriormente, apresentamos aspectos adicionais do tratamento de exceções quando são considerados os requisitos de sensibilidade ao contexto (Capítulo 2). A seguir, apresentamos as principais limitações das abordagens utilizadas atualmente para a implementação do tratamento de exceções. Finalmente, apresentamos uma análise das abordagens apresentadas, tendo em vista o alcance dos requisitos de sensibilidade ao contexto,

usuais em aplicações móveis. O resultado desta análise confirma a necessidade de desenvolvimento de um modelo de tratamento de exceções sensível ao contexto para aplicações móveis.

## 3.1. Aspectos Gerais do Tratamento de Exceções

A análise do tratamento de exceções em sistemas orientados a objetos possibilita a percepção de alguns aspectos fundamentais que devem ser considerados no tratamento de exceções em aplicações móveis sensíveis ao contexto. Por esta razão, apresentamos a seguir alguns destes aspectos gerais, importantes para o projeto de um mecanismo de tratamento de exceções, segundo a taxonomia proposta por Garcia et al. (2001).

### 3.1.1. Representação da Exceção

Para que as exceções sejam utilizadas e processadas internamente pelo sistema de software, elas devem ser representadas internamente. Esta representação pode ser feita através de: (i) símbolos, representação genérica de exceção como, por exemplo, um número inteiro ou uma *string*; (ii) objetos de dados, objeto-exceção que contém os dados referentes à situação excepcional; e (iii) objetos completos, além dos dados e métodos de consulta referentes à situação excepcional, o objeto-exceção provê métodos responsáveis pelo levantamento da exceção.

### 3.1.2. Separação entre Exceção Externa e Interna

Existe uma distinção conceitual entre exceção externa e exceção interna (componente levanta para chamar sua própria medida interna de tratamento), que historicamente se relaciona diretamente às operações `raise` e `signal`, ambas utilizadas para iniciar as atividades tolerantes a falhas por meio de tratamento de exceções (Lee & Anderson, 1990). A operação *signal* permite que um componente "sinalize" uma *exceção externa* para o sistema do qual ele é parte, chamando o tratador internamente ao sistema; *raise* permite que um componente

"levante" uma *exceção interna* para chamar o tratador dentro deste componente. Alguns mecanismos permitem uma clara separação entre exceção externa e interna, enquanto outros não fazem qualquer distinção entre elas. Na linguagem Java, por exemplo, não existe esta distinção. Quando uma exceção é levantada internamente e seu tratador não é encontrado, é feita a propagação automática da exceção para elementos de mais alto-nível.

### 3.1.3. Localização de Tratadores

Tratadores podem ser associados a diferentes regiões protegidas: (i) tratadores de comando, permitem tratadores serem associados com um único comando ou um bloco de comandos em um programa; (ii) tratadores de métodos, quando exceções são levantadas em comandos do método, o tratador ligado a esta exceção é executado; (iii) tratadores de objetos, que definem as medidas para exceções detectadas em uma instância particular de uma classe; (iv) tratadores de classe, permitem a definição de um comportamento excepcional comum para todos os objetos da classe; (v) tratadores gerais[3] de exceções, chamados se nenhum tratador mais específico é encontrado e são válidos em qualquer parte do programa, independentemente de seu atual ponto de execução.

### 3.1.4. Associação de Tratadores

A associação entre uma ocorrência de exceção e seu correspondente tratador pode ser realizada de forma estática, isto é, o tratador é estaticamente ligado a uma região protegida e é usado por todas as ocorrências da exceção durante a execução. Neste caso, não é possível a propagação de exceção e a busca do tratador em tempo de execução. Adicionalmente, a associação pode ser realizada de forma dinâmica, ou seja, a determinação do tratador que deve ser usado para uma ocorrência de exceção é feita em tempo de execução. Existe ainda a associação semidinâmica, que segue um modelo híbrido que combina as duas abordagens anteriores. Podem existir tratadores locais associados estaticamente a

[3] Do inglês, *default*.

certas ocorrências de exceção, porém também é possível que tratadores sejam encontrados dinamicamente em tempo de execução, quando, por exemplo, nenhum tratador é encontrado localmente.

### 3.1.5. Propagação de Exceção

Dizemos que existe propagação de exceção quando o tratador local para uma ocorrência de exceção não é encontrado e a exceção é então propagada para o componente de mais alto nível. Este conceito está fortemente relacionado à ao aspecto de associação de tratadores (Seção 3.1.4). A propagação de exceção pode ser classificada em: (i) propagação automática, a exceção é propagada automaticamente para os componentes de mais alto nível até que o tratador seja encontrado; e (ii) propagação explícita, o tratamento de exceções é limitado ao componente local. Se uma exceção não é tratada no contexto deste componente, então o programa termina ou uma exceção pré-definida geral é propagada automaticamente. O componente local pode opcionalmente propagar uma outra exceção explicitamente.

## 3.2. Aspectos Adicionais Relacionados a Sensibilidade ao Contexto

Devido a suas características de dinamicidade e mobilidade, aplicações móveis baseadas em agentes usualmente devem considerar os contextos onde estão atuando e quais os usuários do sistema onde estão inseridos. Da mesma forma que na execução das atividades normais do sistema, as variações de contextos nas atividades relacionadas ao tratamento de exceções também podem ser utilizadas nas aplicações baseadas em agentes. Por exemplo, pode ser necessário ativar diferentes tratadores para uma mesma exceção levantada em diferentes contextos. Além disso, a estratégia de propagação das exceções pode ser diferente também dependendo do contexto onde uma exceção é levantada.

A seção 3.2.1 descreve uma típica aplicação baseada em agentes sensível ao contexto, que também será utilizada no Capítulo 7 para avaliação da aplicabilidade de nossa proposta para tratamento de exceções sensível ao contexto

em aplicações móveis. A partir da análise de um protótipo desta aplicação que usa MoCA, identificamos um conjunto de requisitos para a tarefa de incorporação de tratamento de exceções sensível ao contexto em aplicações móveis. As seções de 3.2.2 a 3.2.5 apresentam tais requisitos.

### 3.2.1. Estudo de caso: Health Care

Este estudo de caso apresenta o protótipo da aplicação móvel *Health Care* (HC) que oferece suporte ao monitoramento de pacientes com doença cardiovascular sob cuidados médicos. Um conjunto de valores representa o estado do paciente, entre eles a atividade cardíaca destes pacientes, que deve ser continuamente monitorada. Sensores são usados para capturar os sinais vitais de um paciente, como batimento cardíaco, pressão sanguínea, temperatura corporal e freqüência respiratória. Três sensores são colocados no corpo de um paciente. O primeiro sensor monitora a saturação de oxigênio da hemoglobina na pressão arterial e no batimento de pulso. A saturação do oxigênio deve sempre estar acima de 95%. A taxa normal de batimento de pulso em adultos varia de 60 a 100 batimentos por minuto (BPM). O segundo sensor mede a pressão sanguínea do paciente. A pressão sanguínea diastólica varia de 100 a 120 mmHg e a pressão sistólica varia de 60 a 80 mmHg. O terceiro sensor monitora a freqüência respiratória e a temperatura do paciente.

A aplicação HC engloba cenários interessantes de colaboração móvel. Tais cenários podem incluir elementos em diferentes contextos, tais como na região onde se encontra um paciente, ambulância e grupo de suporte à emergência. Várias situações excepcionais podem ocorrer nesta aplicação. Por exemplo, o monitoramento de sensores pode diagnosticar um batimento cardíaco anormal no paciente. Outros exemplos de situações anormais incluem o batimento de pulso fora da faixa normal ou com baixa saturação de oxigênio, que são situações monitoradas por um dos sensores. Um evento pode ser disparado para alertar imediatamente um parente, o médico da família ou o hospital, dependendo da gravidade do problema detectado.

Particularmente, a ocorrência de um evento de ataque cardíaco dispara a colaboração destes elementos em diferentes cenários excepcionais. Por exemplo, o

médico, ao entrar no hospital, pode receber a notificação excepcional e assim decidir-se por auxiliar no tratamento do paciente. Esta é uma situação típica onde subscrições usuais de *middlewares publish-subscribe* (Seção 2.4) funcionam apropriadamente a fim de implementar a notificação da exceção. Entretanto, existem cenários cujos requisitos não são tratados adequadamente apenas pelo uso de *middlewares publish-subscribe*. A próxima seção apresenta as dificuldades encontradas para a incorporação do tratamento de exceções sensível ao contexto na aplicação HC. Cada uma das subseções discute: (i) problemas identificados na implementação de estratégias apropriadas para o tratamento de exceções no estudo HC, (ii) a ilustração destes problemas usando cenários de colaboração móveis concretos do HC, e (iii) as limitações do mecanismo *publish-subscribe* de MoCA em dar suporte a um tratamento de exceções sensível ao contexto em HC.

### 3.2.2. Especificação de Contextos Excepcionais

A aplicação HC especifica diversos "contextos excepcionais" que caracterizam situações em que as variáveis cardíacas coletadas pelos sensores (BPM e pressão sanguínea) excedem seus limites máximo e mínimo. Tais contextos são usados para indicar um problema no sistema cardiovascular de pacientes, como um ataque cardíaco, por exemplo. Contextos excepcionais podem ser entendidos como uma ou mais condições que denotam uma falha ambiental, de hardware ou de software. O tratamento de contextos excepcionais requer um "fluxo de controle excepcional" que é inerentemente diferente do fluxo normal; este último consiste simplesmente em reações baseadas em notificação, enquanto o primeiro requer a propagação de contextos excepcionais aos tratadores apropriados, que também podem ou não ser selecionados dependendo de sua localização física. Por outro lado, a tarefa de propagação de um contexto excepcional pode exigir o envolvimento entre várias entidades, como os parentes do paciente e o grupo de suporte à emergência. Denominamos tal envolvimento entre entidades como sendo uma "colaboração excepcional".

O mecanismo *publish-subscribe* de MoCA não provê suporte direto e efetivo para a implementação de um fluxo de controle excepcional sensível ao contexto. Para *middlewares publish-subscribe*, como o MoCA, que adotam o

modelo de contexto par atributo-valor (Seção 2.3.1), existe a necessidade de subscrições baseadas em expressões regulares que devem registrar explicitamente o interesse por um ou mais contextos. A subscrição normalmente é realizada pelo código nos dispositivos móveis ou *proxies*, que recebem notificações quando as condições de contexto ocorrem de acordo com as circunstâncias variáveis. Entretanto, a especificação de uma situação de contexto excepcional possui uma semântica inerentemente diferente e, como tal, precisa lidar com diferentes elementos em sua especificação, incluindo o escopo de tratamento, tratadores gerais alternativos, tipos de informação contextual que devem ou não ser propagadas junto com a ocorrência de exceção, e assim por diante. Em resumo, em qualquer aplicação móvel sensível ao contexto usando MoCA, subscrições de contexto normais precisam ser definidas de maneira diferente das subscrições de contextos excepcionais.



### 3.2.3. Ausência de Escopos Apropriados para Tratamento de Exceções

Há várias situações na aplicação HC em que o tratamento de exceções requer que vários dispositivos estejam envolvidos dependendo das regiões físicas e outros tipos de informação contextual. Por exemplo, o próprio tratamento das condições excepcionais que envolvem o evento de ataque cardíaco pode requerer que as exceções sejam propagadas a diferentes conjuntos de dispositivos, como aos dispositivos na região onde se encontra a vítima, aos do grupo de suporte à emergência, ou aos dispositivos de parentes ou ao do médico da vítima. Entretanto, a propagação deve ser sensível ao contexto, isto é, a propagação deve considerar a chamada primeiramente aos dispositivos mais próximos à região onde se encontra a vítima de ataque cardíaco. A exceção contextual precisa ser sistematicamente propagada a extensões mais largas até que os tratadores apropriados sejam encontrados.

Por outro lado, é possível que, dependendo da gravidade de uma situação excepcional, o paciente possa especificar que a exceção contextual seja propagada para todas as regiões e grupos de dispositivos móveis envolvidos. Conseqüentemente, as regiões físicas ou um determinado grupo de dispositivos são exemplos de escopos de tratamento contextual que devem ser suportados pelo

*middleware* subjacente. Os próprios tratadores de exceções poderiam ser ativados em todos os dispositivos relevantes de acordo com as preferências do usuário. Contudo, o *middleware* MoCA não suporta tais escopos para tratamento de exceções sensível ao contexto, o que diminui a modularidade do sistema na presença de contextos excepcionais.

### 3.2.4. Necessidade de Tratadores Sensíveis ao Contexto

Existem também os casos onde a seleção dos tratadores de exceções apropriados depende de condições de contexto associadas com os dispositivos envolvidos no tratamento de exceções coordenado. Para a mesma exceção, é necessário criar tratadores para diferentes condições contextuais e garantir que eles sejam corretamente executados. Pode ser necessário utilizar a informação contextual sobre a localização física de um paciente nos tratadores para a situação excepcional do ataque cardíaco de um paciente. Alguns tratadores podem ser selecionados somente se o dispositivo móvel do paciente estiver no contexto de uma região específica, como na sua residência. Novamente, temos que implementar tal controle de tratadores sensíveis ao contexto como parte da aplicação móvel, uma vez que não existe uma facilidade MoCA que dê suporte a este requisito.

### 3.2.5. Tratamento de Exceções Imprevistas

Em uma aplicação móvel aberta, como a aplicação HC, não podemos esperar que todos os dispositivos, nos quais os agentes de software são desenvolvidos por diferentes projetistas, sejam capazes de prever todos os possíveis contextos excepcionais. No caso de HC, por exemplo, durante a ocorrência de um ataque cardíaco, se não existe ninguém em casa além da vítima, um dos vizinhos pode receber a notificação sobre o ataque cardíaco. Entretanto, o dispositivo deste vizinho pode não ser capaz de tratar tal notificação. Assim sendo, é necessário explorar a infra-estrutura de colaboração móvel fornecida por *middlewares publish-subscribe* mesmo na situação de exceções imprevistas.

De fato, dependendo da gravidade de uma exceção, ela deve ser notificada para outros dispositivos móveis mesmo quando eles não registram interesse por um contexto excepcional específico. Em outras palavras, a exceção contextual deve ser proativamente levantada em outros agentes colaborativos e/ou dispositivos móveis pertencentes à mesma região ou escopo. Portanto, aplicações móveis sensíveis ao contexto robustos requerem certa inteligência, isto é, um tratamento de exceções sensível ao contexto proativo. O problema é que modelos de coordenação convencionais, como baseados em espaços de tuplas e na arquitetura *publish-subscribe* como MoCA requerem a subscrição explícita de interesse por parte dos agentes colaboradores.

## 3.3. Limitações de Alguns Trabalhos Relacionados

Existem poucos trabalhos na literatura que consideram questões relacionadas ao tratamento de exceções em aplicações móveis. Nesta seção, analisamos as limitações das três abordagens principais utilizadas para o tratamento de exceções em aplicações móveis, confrontando as facilidades oferecidas por estas abordagens com os requisitos descritos acima para um efetivo suporte à característica de sensibilidade ao contexto nas aplicações. É importante deixar claro que para o escopo deste trabalho, particularmente para a análise desta seção, o termo *agente* se refere especificamente ao conceito de agentes reativos com a capacidade de autonomia de execução, ou seja, eles possuem sua própria *thread* de controle. As aplicações baseadas em agentes exploradas neste trabalho não implicam na presença de aprendizagem e autonomia proativa.

### 3.3.1. Tripathi e Miller

Na abordagem de Tripathi e Miller (2000), as exceções são classificadas em internas e externas ao agente. As exceções internas são tratadas localmente pelo agente e as exceções externas são propagadas para agentes especiais denominados “guardiões”. Os guardiões atuam como tratadores de exceções para um conjunto de agentes e possuem a função principal de monitorar e controlar os agentes da aplicação com relação às condições excepcionais. Desta forma, o modelo de

Tripathi e Miller separa as questões relacionadas ao tratamento de exceções de nível global dos agentes da aplicação e encapsula tais questões nos agentes guardiões. Pode-se dizer que um agente guardião funciona como um tratador associado aos agentes da aplicação e possui uma visão global da situação, pois monitora os agentes que estão cooperando.

De fato, a principal característica da abordagem de Tripathi e Miller (2000) é o suporte ao encapsulamento das atividades de tratamento de exceções em agentes guardiões, com a separação destas atividades do projeto interno dos agentes da aplicação. Contudo, de acordo com Souchon et al. (2003), a abordagem de delegar o tratamento de exceções para agentes guardiões gera alguns problemas, como: (i) os agentes são entidades que integralmente devem encapsular seu próprio comportamento, inclusive o excepcional; (ii) a possibilidade de gerar somente reações gerais para as exceções, devido à impossibilidade de tratadores escritos no contexto da entidade que chama o serviço; (iii) o gargalo criado para o sistema inteiro, causado pela centralização do tratamento de exceções em uma entidade especializada.

Portanto, a principal limitação da abordagem de Tripathi e Miller (2000) diz respeito à impossibilidade de existirem tratadores no contexto das entidades que chamam o serviço. Esta é uma solução extremamente restritiva tendo em vista que a maior parte do código de tratamento de exceções em sistemas reais é específico de aplicação (Garcia et al, 2001). Além disso, cria-se um gargalo para o sistema, pela centralização do tratamento de exceções em uma entidade especializada. Finalmente, o mecanismo de Tripathi e Miller não considera as necessidades de específicas de aplicações móveis sensíveis ao contexto, como a utilização da informação contextual do agente no tratamento de exceções.

### 3.3.2. Souchon, Dony, Urtado e Vauttier

Segundo Souchon et al. (2003), um sistema de tratamento de exceções para aplicações baseadas em agentes deve atender os seguintes requisitos: (i) ser alinhado com as características próprias do paradigma de agentes; (ii) considerar a concorrência e os meios de comunicação assíncrona; (iii) considerar a organização social de agentes (grupos e papéis). A integração do tratamento de exceções com

o paradigma de agentes significa, por exemplo, que as exceções ocorridas entre agentes devem ser executadas como mensagens, ou que tratadores de exceções devem ser associados ao código relacionado às atividades do agente. Além disso, a comunicação assíncrona é considerada um fator importante, pois fornece meios para padrões avançados de execução, uma vez que o emissor de uma requisição continua sua execução normalmente enquanto o receptor ainda está tratando, e, paralelamente, enviando outras requisições de serviço.

Pelo forte acoplamento do tratamento de exceções com o paradigma de agentes, pode-se dizer que a principal característica da abordagem de Souchon et al. (2004) é a associação de tratadores de exceções com as entidades que são responsáveis pela execução das atividades de agentes, isto é, serviços, agentes e papéis. Os tratadores de exceções são estritamente relacionados ao comportamento dos agentes, sendo, portanto, (i) distintos dos tratadores fornecidos pela linguagem de programação, (ii) associados a elementos de execução específicos do paradigma de agentes, (iii) executados pelo levantamento ou propagação de exceções no nível de agentes.

Portanto, pode-se perceber pela descrição acima que a abordagem de Souchon et al. (2004) preocupa-se quase que exclusivamente com o aspecto da localização de tratadores e da definição de regiões de tratadores (como papéis e grupos de agentes), desconsiderando muitos outros aspectos importantes para o desenvolvimento de um mecanismo de tratamento de exceções. Além disso, Souchon et al. não possibilita a associação dinâmica entre tratadores e ocorrências de exceção, o que é fundamental para o tratamento de exceções em aplicações móveis com sensibilidade ao contexto. Por fim, o mecanismo de Souchon et al. não considera a utilização de informações do contexto no tratamento de exceções.

### 3.3.3. Iliasov e Romanovsky

Em Iliasov e Romanovsky (2005a), CAMA é um *framework* para o desenvolvimento de aplicações móveis que fornece um conjunto de abstrações, juntamente com um *middleware* e uma camada de adaptação que permitem aos desenvolvedores tratar características de aplicações móveis como abertura, tolerância a falhas, comunicação assíncrona, como também mobilidade de

dispositivo e de código. Nesta seção, abordamos somente os aspectos de CAMA referentes à tolerância a falhas. Os aspectos referentes à sensibilidade ao contexto foram tratados no Capítulo 2.

CAMA suporta tolerância a falhas através do tratamento de exceções e coordenação de agentes em um ambiente dinâmico aberto. Exceções do espaço de tuplas provêem suporte para tratamento de exceções em um *middleware* de comunicação baseado em Linda (Gelernter, 1985). O principal objetivo é oferecer suporte à propagação de exceções entre agentes móveis que se comunicam através do espaço de tuplas. Para fazer com que o mecanismo de tratamento de exceções de CAMA possa ser considerado universal e portável, assume-se que em grande parte dos sistemas de agentes móveis o espaço de tuplas é o único canal de comunicação entre os agentes e, portanto, o único meio existente para se propagar exceções (Iliasov & Romanovsky, 2005a).

Pode-se dizer então que a principal característica do mecanismo de tratamento de exceções de CAMA é também a sua principal limitação: a abordagem é uma solução específica para aplicações que baseiam suas interações em espaços de tuplas. Adicionalmente, CAMA, assim como as abordagens de Tripathi e Miller (2000) e Souchon et al. (2004), não trata os principais requisitos necessários ao tratamento de exceções sensível ao contexto (Seção 2.5).



## 3.4. Análise das Soluções

Em todas as abordagens descritas na seção anterior, foram analisados os requisitos necessários para a realização de um tratamento de exceções que efetivamente considera a característica de sensibilidade ao contexto. Como já descrito na seção 3.2, entre os requisitos fundamentais de sensibilidade ao contexto, podemos citar a possibilidade de realizar a especificação explícita de contextos excepcionais em uma aplicação, a ativação de tratadores apropriados de acordo com o contexto da aplicação móvel, além da determinação de qual estratégia de tratamento a ser utilizada em um dado momento considerando a informação contextual dos agentes.

Desta forma, como já parcialmente documentado na literatura (Garcia et al., 2005; Iliasov & Romanovsky, 2005b), tanto os modelos tradicionais de tratamento

de exceções como os *middlewares* para desenvolvimento de aplicações sensíveis ao contexto negligenciam os requisitos de sensibilidade ao contexto, não fornecendo soluções apropriadas para os mesmos. Existe, portanto, a necessidade de definição de um modelo para tratamento de exceções sensível ao contexto.