SADE – Sistema de Atendimento de Despacho de Emergências em Santa Catarina

José Roberto C. Perillo[1], João Ricardo Busi da Silva[2], Rodrigo Varga[3] e Eduardo Martins Guerra[1]
[1]Instituto Tecnológico de Aeronáutica – Praça Marechal Eduardo Gomes, 50, Vila das Acácias – CEP 12.228-900, São José dos Campos – SP
[2]Polícia Militar do Estado de Santa Catarina – Rua Visconde de Ouro Preto, 549 – Centro – CEP 88.020-040, Florianópolis – SC
[3]Imagem Soluções de Inteligência Geográfica – Rua Itororó, 555, Vila Bandeirantes – CEP 12.216-440, São José dos Campos – SP

***Resumo* — A criação de sistemas de comando e controle é um desafio hoje enfrentado por diversas instituições, tanto militares quanto civis. Recentemente, foi feito com sucesso o desenvolvimento e implantação de um sistema chamado SADE, para o atendimento de emergências no estado de Santa Catarina. Este artigo tem o objetivo de apresentar detalhes em relação às necessidades e a abordagem para o desenvolvimento desse sistema. Também são apresentadas soluções inovadoras utilizando orientação a aspectos e configuração de metadados específicos de domínio na arquitetura da aplicação. Espera-se que a experiência e lições aprendidas descritas neste estudo de caso possam servir como aprendizado para a implementação de outras aplicações de comando e controle.**

***Palavras-Chave* — Sistema, Integração de Aplicações, Arquitetura de Software, Georreferenciamento.**

I. INTRODUÇÃO

A orientação a aspectos [1][2] é historicamente pouco utilizada em aplicações na indústria. Ela é culturalmente tida como algo "difícil de se aplicar" e como uma tecnologia nova que pode trazer riscos para empresas. Já a utilização de anotações é mais comum, porém anotações de frameworks e APIs, e não anotações referentes ao domínio ou a arquitetura da aplicação.

Anotações [3] são uma forma de definição de metadados, podendo indicar informações adicionais do elemento anotado por ela. Aspectos são módulos de código capazes de modularizar interesses transversais. O uso de anotações com aspectos é mais comum como uma forma de marcar os pontos de junção onde os aspectos irão atuar. Porém, elas também podem ser utilizadas como forma de parametrizar o funcionamento do aspecto para cada elemento interceptado.

Este artigo tem o objetivo de descrever o Sistema de Atendimento e Despacho de Emergências para o estado de Santa Catarina chamado SADE, o qual foi construído sobre uma arquitetura baseada em metadados e aspectos. São apresentadas as circunstâncias que motivaram a criação do sistema, assim como os requisitos considerados para sua construção. Adicionalmente, este artigo também apresenta as inovações do sistema em termos arquiteturais, que proporcionaram flexibilidade para a adição e modificação de componentes transversais e produtividade para a equipe. Mais especificamente, são abordadas questões como o uso de orientação a aspectos, integração com outros sistemas e uso de georreferenciamento. Por fim, o artigo apresenta o estágio atual da implantação do sistema e conclui apresentando os próximos passos.

O artigo está organizado da seguinte forma: a Seção II apresenta o contexto existente antes da criação do SADE; a Seção III descreve o sistema criado assim como sua arquitetura básica; a Seção IV introduz a arquitetura interna do SADE, ressaltando as inovações técnicas utilizadas; a Seção V apresenta como foi feito o uso do georreferenciamento na aplicação; a Seção VI detalha a solução de integração utilizada entre os sistemas envolvidos; e, finalmente, a Seção VII conclui o presente trabalho, ressaltando suas principais contribuições.

II. CONTEXTO DE CRIAÇÃO DO SADE

A Polícia Militar de Santa Catarina (PMSC), visando oferecer ao cidadão catarinense e seus visitantes um melhor atendimento, implantou em 1985 seu primeiro Centro de Operações Policial Militar (COPOM), sediado na Capital do Estado. Nesta ocasião, surgiram os primeiros computadores na corporação. Entre eles, um computador de médio porte denominado SISCO 10.000, com 20 terminais operando em ambiente BioMumps, que correspondia a plataforma básica do aplicativo de Atendimento e Despacho de Ocorrências. A implementação desta solução na PMSC tornou-se um marco histórico. Todavia, com a evolução tecnológica, onze anos após, a equipe técnica da PMSC desenvolveu um novo aplicativo para funcionar em arquitetura cliente/servidor, baseado no sistema operacional Linux e na linguagem de programação SuperMumps. Este aplicativo ficou conhecido na corporação como Estação Multitarefa de Atendimento Policial e Emergências (EMAPE). Muito embora este ainda permaneça em operação nos vinte e sete centros distribuídos no território catarinense, há três anos, estudos vêm sendo conduzidos para oportunizar o desenvolvimento de um novo aplicativo mais apropriado aos novos processos operacionais e as novas tecnologias.

Estes estudos apontaram inúmeras deficiências no atual aplicativo EMAPE e em sua plataforma de operação:

José Roberto C. Perillo, jrcperillo@yahoo.com.br
João Ricardo Busi da Silva, busi@pm.sc.gov.br
Rodrigo Varga, rvarga@img.com.br
Eduardo Martins Guerra, guerraem@gmail.com

- Rotinas sistêmicas (módulos) em desacordo com os procedimentos operacionais utilizados atualmente na PMSC.
- Criação de novas rotinas, como os registros de Boletins de Ocorrências (BO-PM) e de Acidente de Trânsito (BOAT-PM).
- Ausência de tecnologias complementares, como por exemplo:
  - Conectividade com outras plataformas – sistemas de rádio comunicação e de comunicação de dados.
  - Geoprocessamento – localização especializada das ocorrências e das guarnições de serviço.
  - Integração com outros aplicativos – identificação de pessoas e veículos do Sistema Integrado de Segurança Pública (SISP), Polícia Civil (PC), DETRAN e Instituto Geral de Perícia (IGP).
  - Novos módulos de registro e armazenagem de dados.
- Limitações técnicas da linguagem SuperMumps.
- Linguagem de programação descontinuada no mercado – SuperMumps.
- Dificuldades de manutenção do aplicativo, em decorrência das cópias implantadas de forma distribuída no território catarinense, elevando os custos operacionais.
- Constância na saída de técnicos da equipe da PMSC, com conhecimento de programação SuperMumps;
- Escassez de técnicos no mercado com conhecimento em SuperMumps.

Diante desse cenário, no ano de 2008, a PMSC elaborou um projeto em parceria com a Secretaria de Estado da Segurança Pública de Santa Catarina (SSP-SC), com a premissa de buscar recursos financeiros junto a Secretaria Nacional de Segurança Pública (SENASP). A aceitação do Projeto pela SENASP deu-se pela formalização do convênio nº 429/SENASP/2008, oportunizando assim, o desenvolvimento de um novo aplicativo de atendimento e despacho de emergências para as centrais 190, 192 e 193, bem como a inclusão de diversas outras tecnologias, as quais serão detalhadas na Seção III.

III. O SADE

O Sistema de Atendimento e Despacho de Emergências foi criado inicialmente para ser implantando na Central Regional de Emergências de Florianópolis (CREFNS), conforme o convênio com o SENASP. A CREFNS tem em sua área de abrangência sete municípios, considerados a Grande Florianópolis, com uma população aproximada de 875.000 habitantes. Em média, esta central atende 81.000 chamadas telefônicas/mês e realiza atendimentos emergenciais a 14.500 habitantes/mês. Denota-se então a grande importância que o aplicativo SADE tem para os funcionários que atuam diretamente nesta atividade. Por esta razão, a equipe técnica levou um ano e meio para estudar os processos e suas alterações, procedimentos e novas tecnologias, resultando em um artefato denominado de projeto lógico [4], o qual contempla inicialmente os seguintes módulos básicos:

- Módulo de Atendimento: destinado ao registro dos dados do comunicante, como o nome, telefone, endereço do atendimento, fato comunicado e código de atendimento. Tem como funcionários responsáveis os atendentes.
- Módulo de Acompanhamento: destinado a monitorar as solicitações de atendimento em andamento, informando o nº do protocolo, código de atendimento, o fato comunicado, data e hora, agências envolvidas e guarnições empenhadas. Tem como funcionários responsáveis os despachantes.
- Módulo de Despacho: destinado ao empenho de guarnições em um determinado atendimento. Tem como funcionários responsáveis os despachantes.
- Módulo de Homologação: destinado ao registro de homologação do atendimento realizado pelas equipes da Central. Tem como funcionários responsáveis os coordenadores.
- Módulo Administrativo: destinado a manutenção de dados de tabelas do sistema, criação de usuários e controle de acessos. Tem como funcionários responsáveis os membros da chefia e subchefia da CRE.

Para melhor entendimento dos módulos do SADE, é apresentado nas Figuras 1 e 2 o macro fluxo do processo que envolve todas as rotinas da aplicação.

De posse do projeto lógico, a SSP-SC elaborou um edital para contratação de uma empresa para o desenvolvimento do SADE, sendo consagrada como vencedora do certame licitatório a empresa Imagem, a qual iniciou o desenvolvimento em maio de 2010, concluindo em maio de 2011. Coube a Imagem, além do desenvolvimento do SADE, o desenvolvimento do GeoSADE, o I-Gestão e o I-Cidadão, sendo estes últimos aplicativos fundamentados em arquiteturas de sistemas de informações geográficas.

O SADE encontra-se atualmente em fase de implantação na CREFNS, e seu ambiente de produção foi customizado de acordo com a previsão de processamento utilizado pelas aplicações. Testes nos sistemas e no banco de dados foram realizados na fase de homologação e no final do mês de junho foram iniciadas as etapas de treinamentos e operação definitiva da aplicação.

IV. Arquitetura de Software do SADE

Nesta Seção, são apresentadas as características do SADE em termos de arquitetura de software. É apresentada a arquitetura aplicada ao sistema de uma forma macro, as camadas que foram definidas e os requisitos não-funcionais identificados, que foram os requisitos que levaram à escolha da arquitetura aplicada ao SADE. Na Figura 3 é apresentada, de forma macro, a arquitetura do SADE.

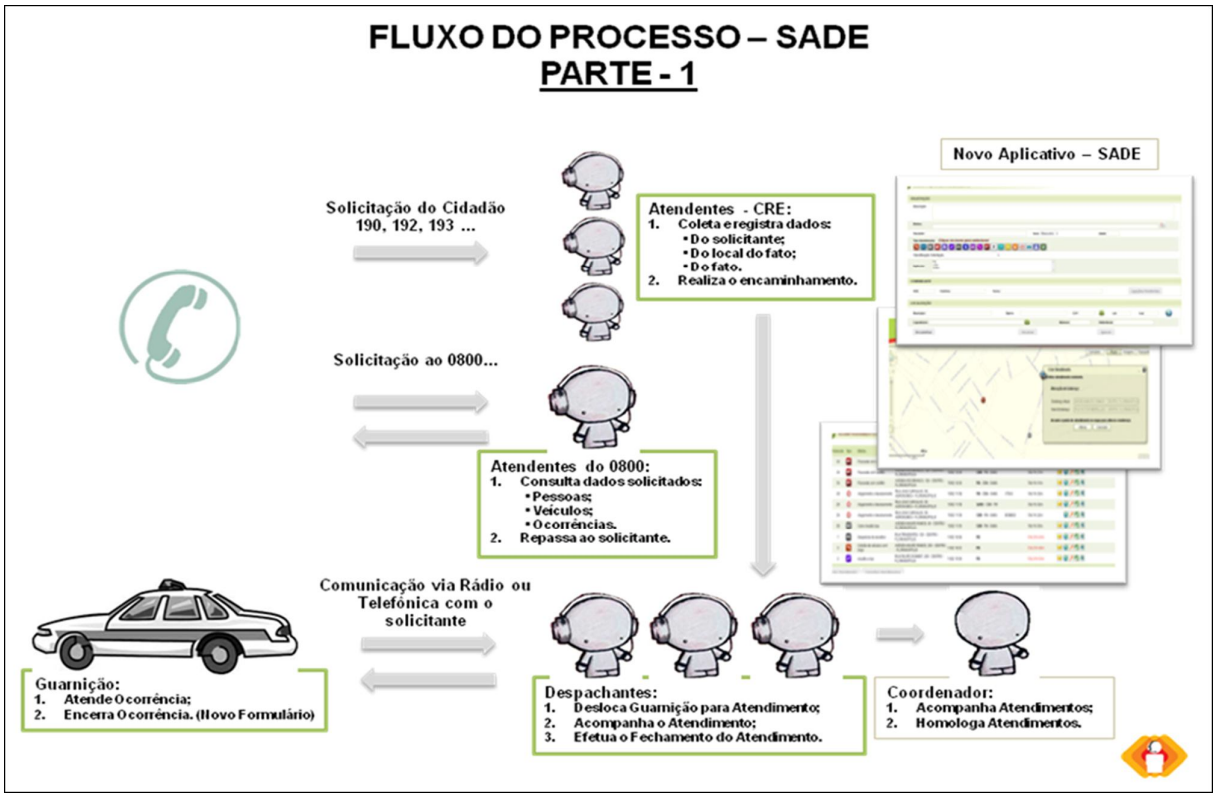

Fig. 1. Primeira parte do fluxo do processo do SADE.

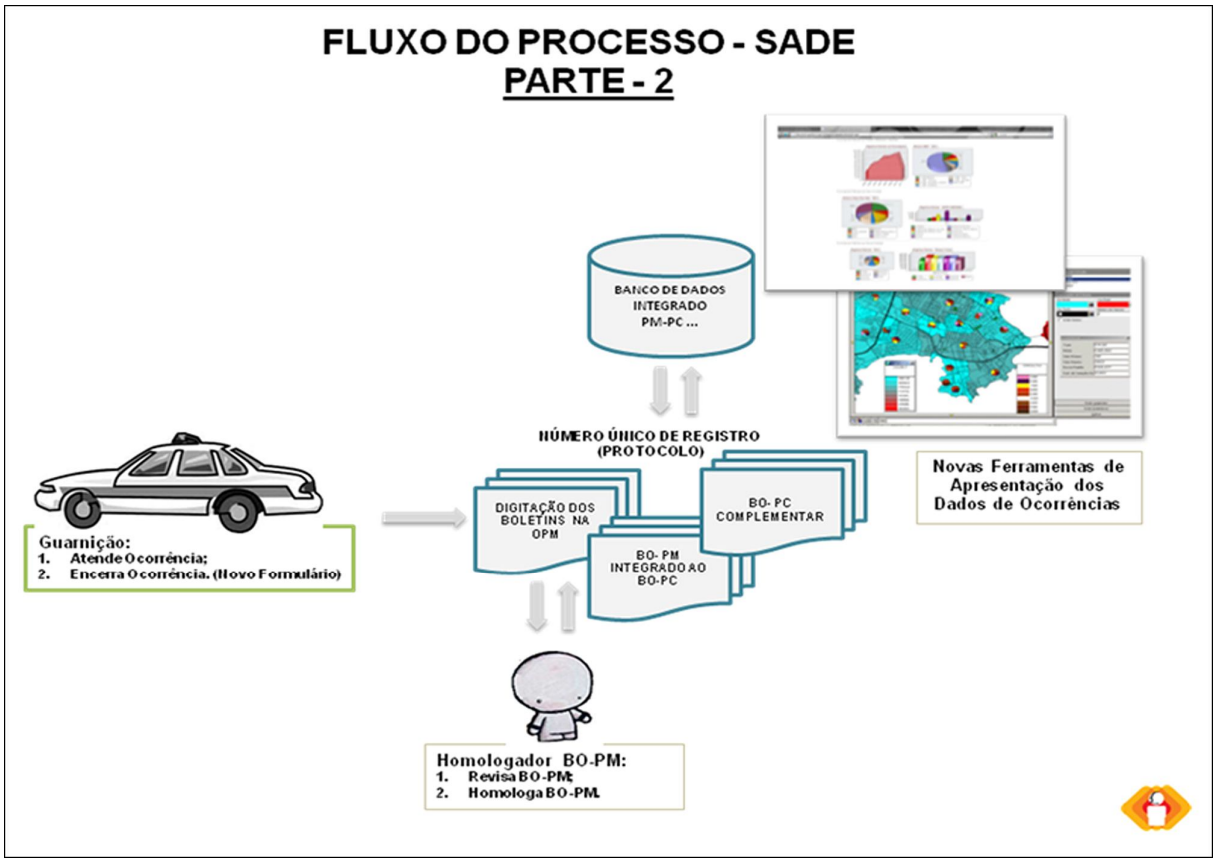

Fig. 2. Segunda parte do fluxo do processo do SADE.

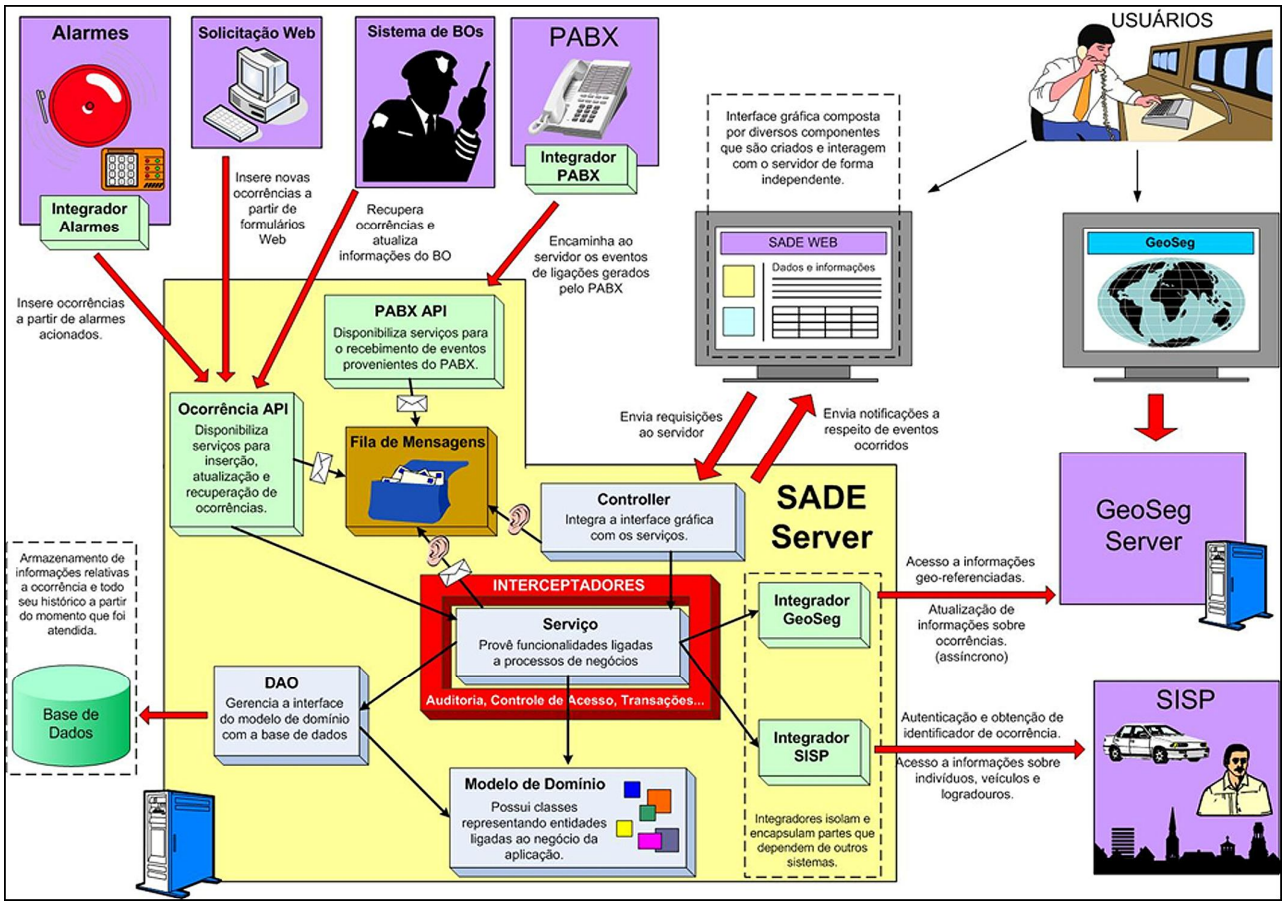

Fig. 3. Arquitetura macro do SADE.

Arquitetura Baseada em Metadados

A arquitetura do SADE baseia-se primariamente em aspectos e anotações. De uma forma geral, aspectos e anotações são historicamente pouco utilizados em aplicações Java. Normalmente, as anotações utilizadas no código pertencem aos frameworks [5] que compõem a arquitetura, e não ao código propriamente dito. Pelo fato de que os aspectos são culturalmente tidos como algo "difícil de aplicar", também são pouco utilizados.

Aspectos são módulos de código capazes de modularizar interesses transversais da aplicação, que normalmente estão relacionados à infraestrutura da aplicação. Por exemplo, em vez de colocar verificações em cada método da aplicação a fim de permitir que somente determinados perfis os acessem, pode-se criar um aspecto que seja disparado a cada chamada de método e efetue esta verificação. Além disso, aspectos são adequados também quando é preciso executar um código ou uma rotina adicional que não faça parte do propósito principal de um método. Por exemplo, suponhamos que, após a execução de um método que faz a reserva de um quarto de hotel, deva-se invocar um web service. Pode-se dizer que a chamada ao web service não faz parte do propósito principal do método, e assim, pode-se criar um aspecto que invoque esse web service e seja disparado após a execução do método que efetua a reserva de quarto de hotel.

Para aplicações Java, anotações são um tipo especial de metadado [6], podendo assim indicar informações adicionais do elemento anotado por ela, sendo assim uma técnica de marcação em nível de programa que permite que o desenvolvedor anote elementos do programa para indicar semântica específica da aplicação ou do domínio [3].

As anotações não afetam diretamente a semântica da classe, uma vez que só adicionam informações complementares que são usadas por frameworks, ferramentas ou bibliotecas de classes. No entanto, isso pode, em tese alterar, a semântica da aplicação como um todo [6]. A linguagem Java oferece a palavra-chave @interface para a definição de uma anotação. Uma anotação não pode conter comportamento, o que significa que, para surtir algum efeito, é preciso que outro código a consuma.

Existem diversas aplicações efetivas para a combinação aspectos e anotações em uma aplicação orientada a objetos. Uma anotação é uma forma de metadado; logo, pode indicar diversas características do elemento que é anotado por ela. Por exemplo, pode indicar que um método deva ser transacional, ter algum nível de segurança, ter logging, etc. Entretanto, em uma aplicação Java, as anotações não podem possuir código, ou seja, precisam que algum código externo as consuma. Isso pode ser feito por um aspecto, o que possibilita que o código anotado não seja invadido com interesses que não digam respeito ao propósito primário do

código. Além disso, definir um aspecto que aja sobre uma determinada anotação permite, por exemplo, que a funcionalidade seja estendida a outras partes do código simplesmente adicionando a anotação onde se queira adicioná-la.

Em uma aplicação orientada a objetos, muitas vezes, um objeto pode requerer o auxílio de outro objeto para executar suas responsabilidades apropriadamente. Nesses casos, uma possível abordagem é instanciar a classe delegada na classe delegadora. O problema dessa abordagem é que, nesse caso, uma classe fica fortemente acoplada à outra. Isso significa que, se a lógica da classe delegada mudar, a classe delegadora possivelmente também terá que mudar. Uma abordagem que permite mais flexibilidade é a injeção de dependência [7], onde um código externo às classes que desempenham esses papéis instancia e arranja esses objetos em tempo de execução. Normalmente, as classes que serão instanciadas e arranjadas são definidas por meio de metadados, o que permite que as implementações possam mudar sem alterar o código das classes.

A abordagem supracitada foi aplicada integralmente ao SADE. Havia alguns requisitos não-funcionais que deveriam ser atendidos, como controle de segurança por tipos de perfil, auditoria de registros no banco de dados, etc. Havia também a necessidade de utilização de Ajax reverso, pois determinados clientes precisariam ser notificados a partir de atualizações no lado do servidor. Para todos os requisitos não-funcionais, havia uma ou mais anotações definidas no código e um aspecto que agia sobre elas, executando a lógica correspondente apropriadamente. Para a definição dos aspectos definidos no SADE, foi utilizado o Spring AOP [8].

Camadas Definidas

Em termos de design, o SADE foi construído basicamente em quatro camadas, mas ao mesmo tempo atendendo ao padrão MVC [9]. As quatro camadas definidas foram apresentação, negócios, persistência e o banco de dados propriamente dito. Além dessas camadas, existem outros componentes satélites, que auxiliam e envolvem as camadas, como aspectos, web services, interceptadores, etc.

Foi utilizado o framework Struts 2 [10] na camada de apresentação, que compreende tanto as páginas, que utilizam suas taglibs, quanto os controladores, que são definidos como Actions. Em alguns casos, foram definidos interceptadores do próprio Struts, que executam alguma lógica antes que os dados enviados pela página executada no navegador no lado cliente cheguem aos controladores. Esses interceptadores, assim como aspectos, também contém lógica que não diz respeito ao propósito dos métodos que recebem os dados nos controladores.

Cada controlador foi definido para atender funcionalidades correlacionadas. Por exemplo, foi definido um controlador para atender as funcionalidades de atendimento, outro para as funcionalidades relacionadas às guarnições, etc. Para se atingir mais reuso dos componentes de negócio, os métodos são finos, contendo estritamente lógica de apresentação. Os dados são enviados para os métodos dos controladores, que repassam o fluxo de execução da aplicação para os componentes de negócio.

Na camada de negócios, foi utilizado o framework Spring [8], que controla o ciclo de vida dos componentes de negócio e cria o arranjo de objetos em tempo de execução. Além disso, ele também foi utilizado para outros propósitos, como definição dos web services, transacionalidade dos métodos de negócios e definição dos aspectos. Os componentes de negócio concentram todas as regras de negócio, o que permite que determinadas funcionalidades sejam expostas a outros tipos de clientes além de navegadores, como clientes de web services, sem necessidade de duplicação de qualquer código.

Na camada de persistência, foi utilizado JPA [11], tendo como framework provedor o Hibernate [12]. Os DAOs [13], que são componentes que lidam diretamente com os componentes do JPA e cuidam dos interesses de persistência, também são controlados pelo Spring e são injetados nos componentes de negócios apropriados.

Em todas as camadas, foram definidos componentes genéricos, que serviram para várias funcionalidades. Por exemplo, foi definido um DAO genérico que contém métodos como save, delete e getAll. Na camada de negócios, foi criado um componente genérico que também contém operações como save, delete e getAll. Nesse componente, o DAO genérico é injetado e cada método que envolve uma transação de banco de dados é anotado com a anotação @Transactional, que informa ao Spring que o método de negócios é transacional, ou seja, corresponde a uma operação ou conjunto de operações que deve ser processado de forma atômica. Os métodos de negócio também recebem anotações de segurança para indicar que o método deve ser protegido, e quando há alteração no estado da aplicação, recebem também a anotação de domínio [6][14][15] @DataModification, que é uma anotação criada na própria aplicação e usada para indicar que os métodos anotados por ela possuem algumas características.

Requisitos Não-Funcionais

Pelo fato de que havia a necessidade de os clientes conectados a aplicação serem notificados da criação de novos atendimentos, foi utilizado Ajax reverso, onde uma conexão fica aberta para cada cliente conectado, e o framework utilizado foi o DWR [16]. A Figura 3 apresenta um mecanismo de fila de mensagens para a notificação da alteração do estado da aplicação aos clientes, que acabou por ser implementado pelo próprio DWR. A partir da criação de um atendimento, a tela de atendimento dos clientes conectados é automaticamente atualizada com o novo atendimento. Além desse caso, alguns outros casos também ativam o Ajax reverso, como na alocação de uma guarnição em um atendimento, ou ao envolver uma agência em um atendimento. Em vez de ter o código de ativação do Ajax reverso do DWR espalhado pelos métodos, foi criado um aspecto que é ativado quando um método com a anotação @DataModification é invocado.

Além disso, essa anotação também indica que as alterações feitas no objeto envolvido no método devem ser

registradas em uma tabela de auditoria de registros no banco de dados. De forma similar ao Ajax reverso, um aspecto é disparado quando um método anotado com essa anotação é invocado e o objeto sendo passado como parâmetro à esse método é comparado com sua versão que já está persistida; caso haja diferenças, é criado um registro de auditoria na tabela de auditorias, indicando quais foram as alterações. Caso seja um registro novo, o registro de auditoria indicará que esse objeto acaba de ser inserido.

No caso de alterações em registros do banco de dados, é necessário comparar o registro corrente com o que está para ser persistido, para que as alterações sejam registradas corretamente. Para essa comparação, foi criado um componente de comparação, que permite verificar todas as alterações feitas em um registro. O aspecto que é disparado quando um método anotado com @DataModification e que cuida dos interesses de auditoria utiliza esse componente, recupera o registro corrente do banco de dados e utiliza o componente de comparação para criar o registro que indica as alterações feitas no registro.

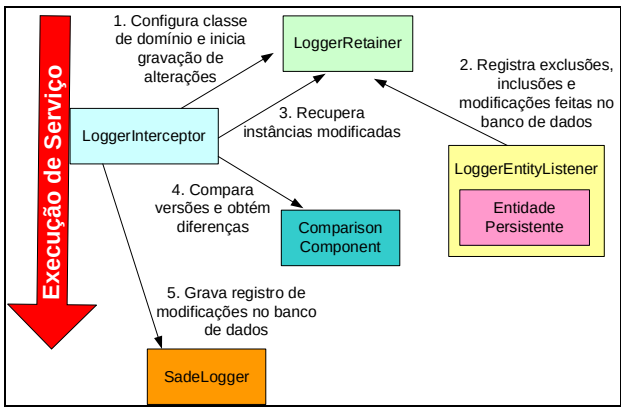

Fig. 4. Funcionamento do componente de logging.

Segurança

Por se tratar de um sistema que lida com informações extremamente sigilosas, foi criado um esquema de segurança próprio para proteger os dados com os quais a aplicação lida, e esse esquema de segurança concentra-se primariamente na camada de negócios da aplicação. Cada classe da camada de negócios é anotada com a anotação @Secured, que também é uma anotação criada na própria aplicação para indicar que os métodos da classe devem ser seguros. Foi definido um aspecto que é disparado a cada vez que um método de uma classe anotado com esta anotação é invocado.

Existem alguns perfis de usuários que utilizam a aplicação, e cada perfil pode acessar um conjunto específico de funcionalidades. Foi criado também um XML que mapeia as funcionalidades e quais perfis tem acesso a estas funcionalidades. Este XML é lido no momento que a aplicação é iniciada e seus dados são alocados em um objeto em memória, e no momento que o usuário se autentica no sistema, o objeto que identifica o usuário na sessão do servidor é preenchido com o conjunto de funcionalidades que corresponde ao seu perfil. A cada requisição que o usuário faz ao servidor, o aspecto de segurança verifica se este usuário tem acesso à funcionalidade sendo requisitada, baseando-se no conjunto de funcionalidades que foi carregado quando o usuário se autenticou na aplicação. Caso não tenha acesso, o usuário é redirecionado para uma página que informa que o acesso foi negado.

Este esquema de segurança foi necessário, pois, como os métodos são genéricos, não é possível mapeá-los de forma estática. Por exemplo, por ser genérico, o mesmo método que salva um atendimento salva também uma guarnição. Assim, um administrador pode salvar uma guarnição, mas não pode criar um atendimento. Em face do tipo de objeto em tempo de execução, o aspecto bloqueia ou não o usuário, baseado no seu perfil.

V. GEORREFERENCIAMENTO

Surgido do sentido de processar dados georreferenciados, o geoprocessamento é conhecido em outras línguas por Geomatic, termo que engloba instrumentos e técnicas para a obtenção de informações espaciais, tais como: processamento digital de imagem, cartografia digital e os sistemas de informações geográficas. [17]

Com a necessidade de identificar espacialmente o local exato de um atendimento e de se buscar os recursos adequados dispersos em uma área geográfica, o SADE apropriou-se dos conceitos e técnicas de um Sistema de Informação Geográfica, originando então o GeoSADE.

Segundo [18], conceitua-se um Sistema de Informação Geográfica (SIG) como sendo um conjunto de mecanismos computacionais que permitem coletar, armazenar e criar análises a partir de uma representação automatizada de dados georrefenciados.

O GeoSADE, módulo que integra o SADE, além de identificar pontos geográficos, áreas de risco, cercas eletrônicas e recursos disponíveis, como guarnições geoposicionadas pelo Sistema de Rastreamento da Frota, permite também consultas a outras fontes de informações geocodificadas, alimentadas por uma outra aplicação de SIG denominada de I-Gestão.

O I-Gestão é uma solução SIG, desenvolvida primeiramente visando à coleta de dados por intermédio de formulários eletrônicos predefinidos. Estes serão preenchidos por policiais militares que atuam na atividade finalística da corporação, e incluem dados do comércio, indústrias, postos de combustíveis, entre outros. Tais dados georreferenciados irão oferecer subsídios informacionais para a tomada de decisão no âmbito operacional e gerencial da corporação. Ressalta-se que os dados criados por intermédio do I-Gestão ficarão disponíveis automaticamente para o GeoSADE.

Num segundo momento, o I-Gestão extrairá dos bancos de dados corporativos, existentes nas diversas áreas do governo, dados das atividades de turismo, educação, saúde, infraestrutura e finanças, permitindo construir representações gráficas e mapas espaciais com o objetivo de também oferecer suporte estratégico às autoridades governamentais.

VI. Integração com Outros Sistemas

Existem alguns sistemas que auxiliam o SADE a cumprir suas responsabilidades, e toda a comunicação entre o SADE e estes sistemas é feita via web services. Por exemplo, a PMSC já possui um cadastro que contém todos os logradouros do estado de Santa Catarina, que é utilizado pelo SADE para validar o logradouro de um atendimento no momento de sua criação. O mesmo acontece com a comunicação do SADE com o GeoSADE, onde este também disponibiliza alguns web services. Assim, o SADE efetua chamadas a estes web services e obtém ou envia dados a outros sistemas.

No caso de algumas regras de negócio, a chamada a estes web services não dizia respeito ao propósito principal do método que implementava estas regras. Foram então definidas algumas anotações de domínio cujo propósito era anotar estes métodos para indicar estas situações. Por exemplo, foi definida a anotação @AlertaGeoSegMudancaEstadoGuarnicao, que indica que o método anotado altera o estado de uma guarnição, como no caso da alocação de uma guarnição em um atendimento, onde a guarnição passa de disponível para alocada. Nesse caso, é necessário comunicar esta mudança de estado ao GeoSADE. Foi definido também um aspecto que é disparado quando um método anotado com esta anotação é invocado, cujo propósito é invocar o web service apropriado do GeoSADE, de forma a comunicá-lo sobre a alteração no estado da guarnição.

O SADE também oferece algumas funcionalidades que podem ser invocadas por meio de web services, e para a criação destes web services, foi utilizado o Spring WS [8]. É possível, por exemplo, criar um atendimento por meio da chamada de um web service passando somente alguns parâmetros. Isto é muito útil no caso de uma emergência, onde o atendimento precisa ser comunicado à polícia de forma rápida. Este web service corresponde à funcionalidade de alarme, ilustrada na Figura 3.

No caso do alarme, existe um cadastro prévio que permite identificar, entre outros dados, o comunicante, a classificação da solicitação, o motivo do alarme, o tipo de atendimento e as agências envolvidas no atendimento. Assim, a partir de dois identificadores enviados na requisição do web service, é possível criar um atendimento de forma rápida e comunicar todas as agências envolvidas no atendimento.

VII. Conclusão

O presente artigo tem como objetivo apresentar a arquitetura baseada em metadados e aspectos que serviu de base para a construção do sistema SADE, criado para o atendimento de emergências no estado de Santa Catarina. Inicialmente o artigo introduz o contexto anterior a implantação do sistema, com as deficiências existentes que motivaram sua criação. Em seguida é mostrada a divisão dos módulos funcionais do sistema, com os principais fluxos suportados pelo mesmo. Depois, a arquitetura do sistema é detalhada, explicitando a divisão de camadas e a solução baseada em metadados e aspectos para a implementação dos requisitos não-funcionais. Adicionalmente, são apresentadas também questões técnicas em relação à solução de georreferenciamento e a integração com outros sistemas.

A principal contribuição desse artigo é deixar registrado o caso de sucesso na criação deste sistema de comando e controle, que pode servir como exemplo para outros que venham a ser construídos. A arquitetura do SADE incorporou técnicas recentes, ainda pouco utilizadas na indústria, para obter flexibilidade para a adição e modificação de componentes transversais e produtividade da equipe. O modelo de integração com outros sistemas e com uma interface georreferenciada, fatores essenciais na criação de sistemas contemporâneos de comando e controle, também podem ser utilizados como base para arquiteturas de outros sistemas similares.

Como trabalhos futuros, uma possibilidade dentro da arquitetura do SADE seria distribuí-lo de forma a permitir que cada central possuísse seu servidor e o mesmo interoperasse com os servidores de outras centrais. Essa arquitetura permitiria autonomia de cada central no caso de uma excepcional queda nas comunicações. Outro trabalho futuro seria generalizar as soluções implementadas no SADE para um framework mais geral, de forma que elas possam ser utilizadas mais facilmente em outros sistemas.

Referências

[1] FILMAN, Robert; FRIEDMAN, Daniel. Aspect-oriented programming is quantication and obliviousness. In: WORKSHOP ON ADVANCED SEPARATION OF CONCERNS AT OOPSLA, Oct. 2000. Proceedings… Mountain View: RIACS, 2000. (RIACS Technical Report, 01.12).

[2] SILVA, Jefferson. Frameworks orientados a aspectos baseados em metadados. São José dos Campos: Instituto Tecnológico de Aeronáutica, 2008.

[3] ROUVOY, Romain; MERLE, Philippe. Leveraging component-oriented programming with attribute-oriented programming. IN: INTERNATIONAL ECOOP WORKSHOP ON COMPONENT-ORIENTED PROGRAMMING, 11., 2006, Nantes. Proceedings...[S.1: s.n], 2006.

[4] Projeto Lógico. Projeto do sistema de atendimento e despacho de emergências. PR010-2007. Centro de Comunicações e Informática – PMSC, dezembro 2007.

[5] GUERRA, Eduardo. A conceptual model for metadata-based frameworks. São José dos Campos: Instituto Tecnológico de Aeronáutica, 2011.

[6] PERILLO, José Roberto Campos; GUERRA, Eduardo; FERNANDES, Clovis. Daileon: A tool for enabling domain annotations. In: WORKSHOP ON REFLECTION, AOP AND META-DATA FOR SOFTWARE EVOLUTION, 6., 2009. Proceedings…[S.l:s.n.], 2009.

[7] FOWLER, Martin. Inversion of control containers and the dependency injection pattern. Disponível em: <http://martinfowler.com/articles/injection.html>. Acesso em: 10/07/2011.

[8] WALLS, Craig; BREIDENBACH, Ryan. Spring in action. 2. ed. Greenwich: Manning Publ.; 2007.

[9] FOWLER, Martin. Patterns of enterprise application architecture. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[10] BROWN, Don; MICHAEL DAVIS, Chad; STANLICK, Scott. Struts 2 in action. Greenwich: Manning Publ., 2008.

[11] Pro JPA 2: Mastering the Java(TM) Persistence API (Expert's Voice in Java Technology). Apress, 2009.

[12] BAUER, Christian; KING, Gavin. Java persistence with hibernate. Greenwich: Manning Publ., 2006.

[13] ALUR, Deepak; MALKS, Dan; CRUPI, John. Core J2EE patterns: best practices and design strategies. prentice hall PTR, 2. ed. Upper Saddle River : Prentice Hall 2003.

[14] DOERNENBURG, Erick. Domain annotations. In: The Thoughtworks Anthology: Essays on Software Technology and Innovation. Raleigh: Pragmatic Bookshelf, Mar.2008. Chapter 10, p. 121-141.

[15] PERILLO, José Roberto Campos et al. Metadata modularization using domain annotations. In: WORKSHOP ON ASSESSMENT OF CONTEMPORARY MODULARIZATION TECHNIQUES (ACoM.09) at OOPSLA, 3., 2009 Orlando. Proceedings…[S.l:s.n.], 2009.

[16] DWR – Easy ajax for java. Disponível em: <http://directwebremoting.org>. Acesso em: 10/07/2011.

[17] MOURA, Ana Clara Mourão. Geoprocessamento na gestão e planejamento urbano. Belo Horizonte: Ed. da Autora, 2003.

[18] ROCHA, Cézar Henrique Barra. Geoprocessamento: tecnologia transdisciplinar. Juiz de Fora, MG: Ed. do Autor, 2000.