SIPLOM 3: A Terceira Geração do Sistema de Comando e Controle da Defesa

Leandro Ouriques Mendes de Carvalho, Rodrigo Abrunhosa Collazo, Aldo Dolorico Balbi, Bruno Hansen Dias
Centro de Análise de Sistemas Navais (CASNAV), Praça Barão de Ladário, s/nº - Ilha das Cobras, Rua da Ponte, Ed. nº 23 do AMRJ
Centro – Rio de Janeiro – RJ – Brasil – CEP 20091-000

***Resumo* — O Sistema de Planejamento Operacional Militar (SIPLOM) representa o componente de software de Comando e Controle do Ministério da Defesa. A sua terceira versão, feita ao longo de 2011, representa um ponto de inflexão no seu desenvolvimento. Nela, a arquitetura foi centralizada e reduzida em número de módulos, o modelo JC3IEDM foi incorporado ao modelo de dados, e a interação do usuário com o sistema foi bastante simplificada. Desta forma, diversos desafios tecnológicos foram superados, como o uso da linguagem PHP e Java em um mesmo sistema. Assim sendo, este artigo tem por objetivo apresentar a arquitetura do SIPLOM 3, discutindo as principais soluções tecnológicas implementadas para viabilizar o atendimento dos requisitos de alto nível estabelecidos pelo Ministério da Defesa (MD).**

***Palavras-Chave* — Comando e Controle, Arquitetura, SIPLOM**

I. INTRODUÇÃO

A rápida transformação dos processos políticos e sociais dos dias atuais tem impactos decisivos nos Sistemas Militares de Comando e Controle, que precisam constantemente aperfeiçoar suas doutrinas, procedimentos e normas e garantir o contínuo aperfeiçoamento do seu pessoal.

Com o avanço das tecnologias da informação, o aprofundamento das relações econômicas, financeiras e comerciais, transnacionais e internacionais, e o crescente enfoque multilateral em questões socioambientais, animosidades e consequentes situações de crise entre Estados na cena internacional têm se tornado mais frequentes, apesar da proscrição da guerra como ferramenta política a luz do direito internacional. Neste cenário, torna-se mister que os decisores político-militares sejam capazes de acelerar o seu ciclo de tomada de decisão, de forma a maximizar as oportunidades que se colocam e minimizar as eventuais fraquezas do poder nacional.

Neste particular, o investimento em tecnologia militar, principalmente associada a ferramentas de software que permitam abreviar o tempo de conscientização situacional em alto grau de abstração, compatível com o nível político-estratégico, e fornecer recursos de auxílio à tomada de decisão, é essencial para suportar os processos organizacionais e doutrinários inerentes ao espectro da Defesa do país.

Consciente desta necessidade de evolução tecnológica, estabelecida na Estratégia Nacional de Defesa [1] e na Doutrina Militar de Defesa [2], o Ministério da Defesa (MD) tem investido no desenvolvimento continuado do seu Sistema de Comando e Controle, o Sistema de Planejamento Operacional Militar (SIPLOM), em parceria com o Centro de Análises de Sistemas Navais (CASNAV) desde 2004.

Desenvolvido ao longo de 2011, o SIPLOM 3.0 caracteriza-se pelo avanço em quatro vetores tecnológicos em relação às versões anteriores: desenvolvimento do Módulo de Gerência de Dados (MGD) com tecnologia web, permitindo o acesso ao sistema de qualquer computador e independente de plataforma; adoção de uma arquitetura centralizada, permitindo a criação de um banco de dados único e, consequentemente, eliminando a necessidade de exportações de dados entre os módulos dos sistemas; melhorias funcionais no Módulo de Apresentação de Forças (MAF), permitindo o desenho de elementos de controle e planejamento de deslocamentos de meios diretamente no ambiente georreferenciado do MAF; e aderência ao modelo de dados JC3IEDM.

Este artigo tem por objetivo apresentar a arquitetura do SIPLOM 3, analisando as principais tecnologias empregadas na busca de soluções práticas para os requisitos de alto nível definidos pelo MD. Para isto, a seção seguinte oferece uma visão geral do sistema, por meio de uma breve retrospectiva histórica de seu desenvolvimento. A seção 3 introduz a estrutura arquitetural do sistema e seus principais módulos. As seções 4 e 5 discutem com maior detalhamento técnico as soluções encontradas para a arquitetura do servidor e a arquitetura do cliente, respectivamente. Por fim, na seção 6, são feitas as considerações finais e esboçados alguns trabalhos futuros.

II. SIPLOM

O SIPLOM sempre se caracterizou por ser um sistema de arquitetura cliente-servidor. Inicialmente, seus clientes eram softwares desenvolvidos em Delphi, que eram instalados e executados em sistemas operacionais Windows. Os servidores eram representados pelos bancos de dados que estavam localizados no MD e nos Centros de Comando e Controle (C2) das Forças. Eles executavam o sistema gerenciador de banco de dados (SGBD) Firebird.

Inicialmente o SIPLOM era composto por muitos módulos. Ao longo dos dez anos de projeto, seguindo a tendência de desenvolvimento de sistemas, seus principais módulos evoluíram para uma arquitetura web. Estes módulos desempenhavam os principais objetivos do sistema que eram o cadastro dos acompanhamentos e a apresentação do Teatro Operações no mapa. Os outros módulos que forneciam apoio aos módulos principais foram sendo descontinuados por mudanças da arquitetura do sistema (por exemplo, o módulo

de gerência de eventos para replicação de dados e o módulo de gerência de mídias) ou foram substituídos por ferramentas *open source* já consolidadas (por exemplo, os módulos de webmail e de chat). Além disso, o banco de dados Firebird foi aos poucos substituído pelo PostgreSQL uma vez que não dispunha de recursos geoespaciais.

Se por um lado, a arquitetura do SIPLOM se tornou mais enxuta, o seu modelo de dados foi se expandido continuamente. O ápice dessa tendência ocorreu exatamente na atual versão do sistema, quando o mesmo se tornou aderente ao modelo JC3IEDM (*Joint Consultation, Command & Control Information Exchange Data Model*), ampliando o volume de informações gerenciadas pelo sistema para atender as necessidades de Comando e Controle (C2) da Defesa. O JC3IEDM é um modelo de intercâmbio de informações proposto pelo *Multilateral Interoperability Programme* (MIP) [3], cuja finalidade é promover, desenvolver e aprimorar especificações de interfaces para reduzir a lacuna de interoperabilidade existente entre Sistemas de Comando e Controle. No SIPLOM, ele foi adaptado para a estruturação do próprio banco de dados, por meio de extensões de objetos, tipos e classes, e construção de interfaces que filtram as complexidades relacionais deste modelo para os usuários.

III. ARQUITETURA E MÓDULOS DO SISTEMA

Atualmente, o SIPLOM é um sistema cliente-servidor com arquitetura inteiramente web, constituído por três módulos principais: Módulo de Gerência de Dados (MGD), Módulo de Apresentação das Forças (MAF) [4] e Módulo de Intercâmbio de Dados (MID).

A Fig. 1 apresenta a arquitetura atual do sistema. Em uma visão em alto nível, ela mostra a camada do servidor, a interface do cliente e um agente externo que exporta dados para o SIPLOM. Os módulos MGD, MAF e MID estão respectivamente destacados nas cores: verde, azul e laranja.

A camada dos servidores é constituída de três máquinas virtuais que contém respectivamente: o servidor de banco de dados, o servidor web que disponibiliza o acesso ao MGD e ao MAF e o servidor de aplicação que hospeda o MID.

O MID é um serviço que fica permanentemente em execução e não possui interface com o usuário. Ele provê a interoperabilidade entre o SIPLOM e o Sistema Naval de Comando e Controle (SISNC2) da Marinha, uma vez que importa os dados do Sistema de Apresentação Gráfica e Bancos de Dados (SAGBD) referentes a informações e localização de portos e cidades; informações, mídias, posicionamento e situação dos meios navais e dos navios mercantes; e informações e mídias das operações em que os meios navais estão empregados.

O MGD é um módulo de cadastro das principais informações do sistema: tipos, meios, operações, comandos operacionais, localizações, documentos, mídias, capacidades operacionais; através do qual elas são armazenadas no banco de dados do sistema. Estas informações são apresentadas em tabelas dinâmicas que realizam operações de ordenação e filtro de dados, permitindo os usuários montarem visões bem particulares dos dados.

O MAF é o módulo onde as informações cadastradas no MGD são apresentadas georreferenciadas em um mapa, oferecendo aos tomadores de decisão a consciência situacional do Teatro de Operações. Ele possui as funcionalidades básicas de um sistema de informação geográfica: mover o mapa (pan), alterar a escala do mapa (zoom) e identificar os elementos apresentados no mapa (info) como acompanhamentos, planejamentos ou feições geográficas. Ele também possui muitas funcionalidades que permitem ao usuário configurar as diversas cartas e imagens que compõem o mapa para montar um cenário ou salvar uma sequência de cenários que irão compor um apresentador para um *briefing*. Estes cenários são gerados a partir de uma janela de tempo definida pelo usuário onde ele também seleciona as Forças que serão exibidas. O MAF ainda possui recursos para medir distâncias, exibir o alcance dos sensores e armamentos, implementar consultas espaciais e filtros nos dados, e definir a projeção do mapa.

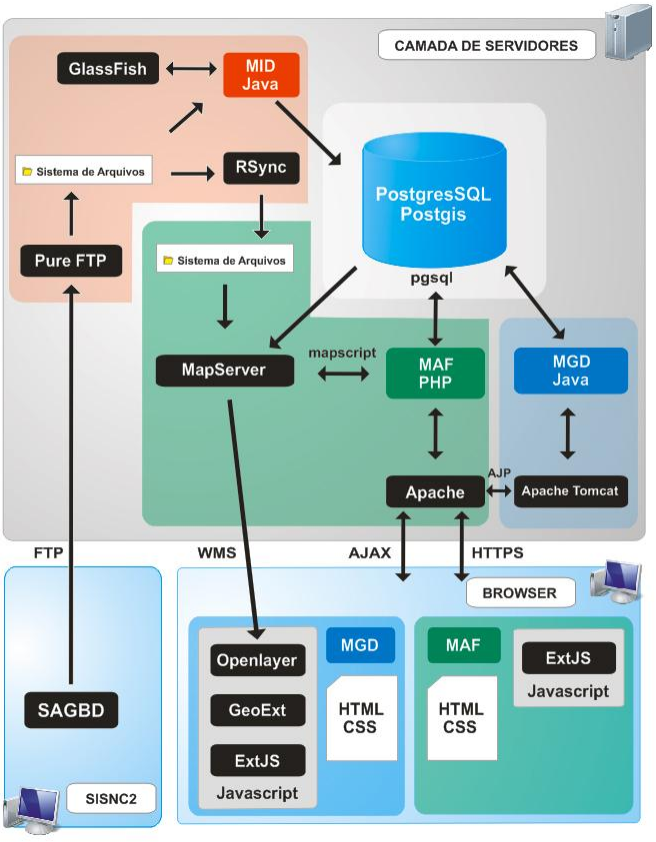

Fig. 1. Arquitetura do SIPLOM.

O MGD e o MAF são sistemas web acessados pelos usuários através de um navegador web (browser), preferencialmente o Firefox. Seguindo as solicitações de segurança do MD, a comunicação desses dois módulos entre o cliente e o servidor é estabelecida através do protocolo HTTPS para garantir que as informações do sistema trafeguem em modo seguro.

IV. ARQUITETURA DO SERVIDOR

Na arquitetura atual, os bancos de dados foram centralizados em um único servidor no MD. Este servidor é uma máquina dedicada à execução exclusiva do SGBD PostgreSQL, que armazena a base de dados única acessada por todos os módulos do SIPLOM.

O PostgreSQL [5] é um dos SGBDs de código aberto mais avançados e possui recursos como integridade transacional, controle de concorrência e suporte a linguagens procedurais. Ele possui ainda uma extensão chamada PostGIS [6] que permite armazenar objetos geográficos e fazer consultas geoespaciais. O PostGIS define três novos tipos de dados: ponto, linha, polígono e também possui um conjunto de funções que realizam os cálculos geométricos e analisam as relações entre os objetos geográficos.

O acesso do usuário ao MGD e ao MAF é feito através de uma interface única de login. O módulo que será inicialmente aberto é definido de acordo com a preferência do usuário. A porta de comunicação do sistema no lado do servidor é representada pelo servidor web Apache [7]. As requisições provenientes do MAF apontam para uma *url* que representa um *script* em PHP, enquanto que as requisições do MGD apontam para um *servlet* Java, contudo o Apache não interpreta Java. Para definir qual caminho a requisição tem que seguir dentro do servidor, o Apache é configurado com um módulo que permite criar um *proxy* com outro servidor web. O servidor Apache Tomcat [8] é o responsável por interpretar as requisições Java. Deste modo, quando o Apache recebe uma requisição, ele primeiramente a analisa. Então, se ela apontar para um *servlet/JSP*, o Apache a redireciona para o Apache Tomcat Servlet/JSP container através de um conector que utiliza um protocolo AJP (*Apache JServ Protocol*).

As requisições do MGD estão sempre atualizando as informações do sistema através das seguintes ações: incluir, alterar, excluir ou consultar. Para atender a essas ações, as requisições foram implementadas seguindo os princípios do *REST* (*Representational State Transfer*) [9] que representa um estilo de arquitetura em que cada mensagem HTTP possui uma *url* distinta, contendo toda a informação necessária para a compreensão da requisição. Ele usa os métodos *GET*, *POST*, *PUT* e *DELETE* já existentes no protocolo HTTP, que refletem respectivamente as ações de consultar, incluir, alterar e excluir. O conteúdo dessas requisições é salvo em objetos que são instâncias das classes de domínio do SIPLOM. Esses objetos são persistidos nas tabelas do banco de dados do SIPLOM, através de um mapeamento objeto-relacional implementado pelo framework *Hibernate* [10] utilizando anotações no padrão JPA [11] e respeitando a integridade referencial.

As requisições do MAF recebidas pelo servidor Apache são encaminhadas para o interpretador do PHP [12]. Algumas delas exigem acesso ao banco de dados do sistema que o PHP realiza através de sua biblioteca pgsql. Entretanto, a maioria das requisições tem como consequência a atualização do mapa. O MapServer [13] é a *engine* responsável pela geração dos mapas no MAF.

O MapServer surgiu de um projeto de código aberto e tem como finalidade exibir mapas dinâmicos através de um navegador web. A descrição do mapa é especificada em um arquivo texto estruturado chamado *mapfile* que é interpretado pelo MapServer para gerar o mapa. O MapServer pode ser executado como uma CGI (*Common Gateway Interface*) ou através de uma API (*Application Programming Interface*), que possibilita que o *mapfile* seja manipulado dinamicamente.

O PHP acessa o MapServer através da API MapScript que permite a criação e atualização dinâmica do *mapfile*. O *mapfile* define o tamanho e a extensão da área do mapa, o tipo de imagem que será gerada, lista as camadas que constituirão o mapa, indicando as respectivas fontes de dados, projeções, simbologia, entre outros. As fontes de dados constituem arquivos vetoriais: shapefiles, DGN, GPX; arquivos raster: BMP, JPG, PNG, KAP e TIFF; dados georreferenciados em um banco de dados e serviços web [14] especificados pelo *Open Geospatial Consortium* (OGC) [15]: WMS, WFS.

Como foi apresentado na seção II, o SIPLOM se comunica com o SISNC2 da Marinha através do MID. O servidor do MID disponibiliza um FTP através do aplicativo PureFTP. O SAGBD acessa este serviço e envia um arquivo ZIP compactado e criptografado que contém um arquivo de dados no formato XML e arquivos de mídias dos meios navais e operações. O arquivo criptografado é salvo no sistema de arquivos do servidor.

O MID foi desenvolvido em Java e necessita de um servidor de aplicação para executá-lo. Esta função é desempenhada pelo Glassfish [16], que é um servidor *open source* desenvolvido pela Sun e que oferece suporte às especificações da plataforma Java EE (Entreprise Edition) para sistemas distribuídos ou multicamadas.

Primeiramente, o MID decodifica o arquivo ZIP encriptado. Em seguida, o conteúdo do arquivo é decomposto em objetos que são instâncias das mesmas classes de domínio do SIPLOM utilizadas pelo MGD. O MID também utiliza o mesmo procedimento de mapeamento objeto-relacional para persistir o conteúdo dos objetos no banco de dados.

As mídias são copiadas para o sistema de arquivos do servidor onde estão localizados o MAF e o MGD. Há um controle para evitar que as mídias já copiadas sejam reenviadas. O software RSync é o responsável por este sincronismo entre os servidores e envia apenas os arquivos necessários. Estas mídias são acessadas pelo MAF que exibe, por exemplo, as fotos de um meio identificado no mapa.

V. ARQUITETURA DO CLIENTE

Conforme já apresentado na seção III, o lado do cliente da arquitetura do sistema é representado pelo navegador (*browser*) através do qual os usuários acessam o SIPLOM. De acordo com a especificação de requisitos do SIPLOM, o Firefox é o *browser* que deve ser utilizado para acessar o sistema, contudo, ele já teve seu comportamento garantido em outros *browsers* como Chrome, Safari e Opera. O browser é um programa que interpreta vários tipos de

arquivos (HTML, XML, JPEG, GIF, PNG, etc.). Contudo, a única linguagem de programação que ele interpreta é o *Javascript*. As subseções a seguir apresentam como a evolução do *Javascript* refletiu nas decisões tomadas durante o desenvolvimento do SIPLOM, incluindo a utilização do framework Ext JS e das bibliotecas GeoExt e OpenLayers.

V.1. Javascript

Na versão anterior, o SIPLOM iniciou sua evolução para um paradigma web. Ele era composto do lado do cliente essencialmente por páginas HTMLs dinâmicas geradas pelo PHP no servidor. Estes arquivos HTML faziam referência a folhas de estilo CSS utilizadas para separar o formato do conteúdo no desenvolvimento das páginas. Nesse processo, o *Javascript* foi um recurso que teve papel fundamental para a consolidação do sistema no paradigma web. Ele é uma linguagem em forma de *script*, interpretada pelo navegador e utilizado, sobretudo, para aperfeiçoar a interatividade do usuário com o aplicativo.

Na versão atual, toda a navegação do usuário dentro de cada módulo do sistema acontece praticamente dentro de um único documento, que é modificado dinamicamente a cada comando. Não é mais necessário trocar o endereço das páginas. A adoção deste modelo de navegação foi possível pelo aperfeiçoamento do Javascript em dois sentidos. O primeiro foi o suporte ao padrão DOM [17] definido pelo W3C [18], permitindo alterar e editar a estrutura, conteúdo e estilo de um documento eletrônico. A segunda foi a sua utilização em conjunto com o XML, empregando uma técnica conhecida como AJAX (Asynchronous JavaScript and XML). O AJAX é implementado com o objeto XMLHttpRequest que permite conectar de forma assíncrona (independente da linearidade da execução do *script*) ao servidor e recuperar somente os dados requisitados. Apenas o que é de interesse do usuário trafega entre o cliente e o servidor, não havendo mais necessidade de recarregar todo o conteúdo, o que significa um grande ganho de desempenho do aplicativo.

Outro aprimoramento do Javascript foi o modo de utilização com o JSON (Javascript Object Notation) [19], que é um formato para intercâmbio de dados computacionais mais leve que o XML. Através dele o *Javascript* pode ser construído de forma orientada a objeto. Tanto os dados trocados com o servidor, como os manipulados no lado cliente são organizados em objetos.

Após o surgimento desses recursos, o *Javascript* atingiu um novo patamar no desenvolvimento de sistemas web. Surgiram várias bibliotecas para auxiliar o seu desenvolvimento, das quais podemos destacar o Jquery [20] e o Prototype [21], que possuem vários componentes de navegação e animação que melhoram a usabilidade dos sistemas. Estas bibliotecas tiveram muita receptividade junto aos desenvolvedores de sistemas.

O próximo passo da evolução da utilização do *Javascript* se deu através da criação de verdadeiros *frameworks* de desenvolvimento, como por exemplo, o Google Web Toolkit (GWT) e o Ext JS. Eles contribuíram para que a camada de apresentação dos sistemas migrasse para o lado do cliente; anteriormente esta camada de apresentação era gerada pelas linguagens de programação do lado do servidor, como PHP e Java. Estes *frameworks* são classificados como ferramentas RIA (*Rich Internet Aplication*), porque efetivamente enriquecem a interface dos sistemas web ao trazerem as características e funcionalidades dos tradicionais sistemas *desktop* para a web.

V.2. Ext JS

O Ext JS [22] é um framework desenvolvido pela empresa Sencha que pode ser utilizado livremente através de uma licença GPL. Ele possui coleção de componentes complexos e extensíveis, que foram construídos seguindo o paradigma de orientação a objetos (OO), possuindo, portanto, atributos e métodos configuráveis. Muitos dos componentes que antes eram confeccionados pela equipe do próprio SIPLOM nos módulos MGM e MAF foram integrados com os objetos do Ext JS potencializando a interatividade e aumentando a qualidade do sistema. O tempo que antes era dedicado ao desenvolvimento destes componentes passou a ser dedicado à composição de componentes especializados ao negócio e à criação de novas interfaces mais inteligentes. Outra característica muito marcante é a sua compatibilidade entre os principais navegadores disponíveis no mercado: Firefox, Chrome, Safari, Internet Explorer e Opera. É realmente possível desenvolver a interface de um sistema sem a necessidade de se preocupar em fazer ajustes para adequá-lo às particularidades de cada navegador.

O Ext JS permitiu que a camada de apresentação do MGD fosse projetada para ser executada somente pelo *browser*. Ele renderiza todos os objetos gráficos, caracterizando uma visão do lado do cliente sob a ótica do padrão de arquitetura MVC (*Model-View-Controller*) [23]. Nenhum componente da camada do lado do servidor se responsabiliza pelo trabalho de configuração e construção destes objetos, que poderia ser desenvolvido em PHP, ou Java/JSF. A Fig. 2 apresenta um formulário de cadastro do MGD onde alguns componentes podem ser identificados (por exemplo, menu, *grids*, janelas flutuantes, painéis, abas), ilustrando a aparência de sistema desktop do MGD. O Ext JS também possui outros componentes que promovem a interação do usuário, mas não são visíveis para ele: *stores* para armazenar os dados exibidos nas *grids* e nos combos; controles de validação e de comunicação com o servidor; e outros.

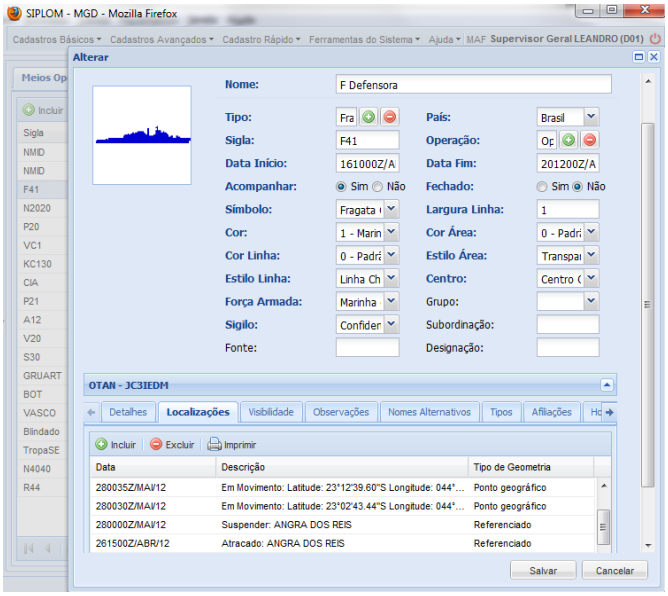

Fig. 2.Interface do MGD.

A comunicação entre cliente e servidor é feita a partir de requisições AJAX que só trafegam dados no formato JSON e que obedecem aos princípios de uma arquitetura REST. No corpo da requisição não há declarações de tags HTML que representaria atualização do DOM da página. Então, quando o cliente recebe um conjunto de objetos ou de registros de uma tabela, os componentes do Ext JS geram o HTML dinamicamente sob demanda.

A interface do MAF não sofreu mudanças radicais, entretanto, também foi evoluída para utilizar alguns recursos do Ext JS que oferecem uma melhor usabilidade para o usuário. A árvore de Operações passou a ser montada utilizando a árvore de estrutura de dados do Ext JS que melhor gerencia uma estrutura com grande número de nós. O menu foi substituído pelo objeto menu do Ext JS porque é mais simples de ser configurado e mantido pelo desenvolvedores.

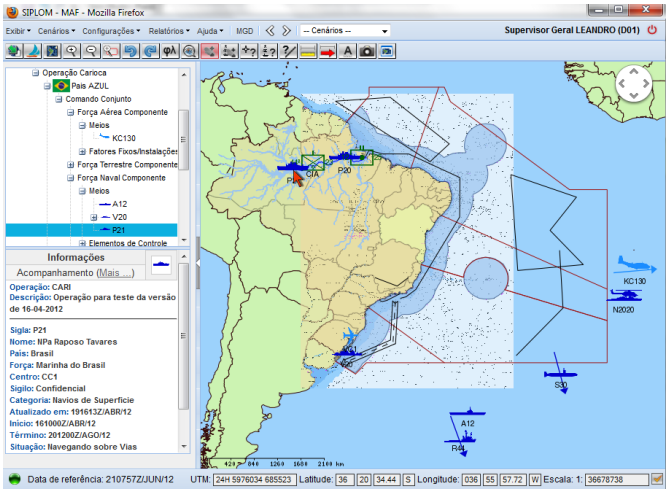

Fig. 3. Interface do MAF.

A Fig. 3 ilustra a nova interface do MAF que se aproxima do padrão utilizado no MGD. Utilizou-se a folha de estilos CSS do Ext JS uma vez que os tons de azul claro que predominava na interface eram mais agradáveis à visão segundo os usuários e suas classes poderiam ser referenciadas pelo outros componentes do Ext JS que estavam sendo utilizados.

V.3. OpenLayers

O cadastro de acompanhamentos no MGD exige que o usuário informe a localização dos mesmos. Na versão anterior do SIPLOM, o cadastro da localização do MGD era muito burocrático e suscetível a erros, porque os usuários tinham que digitar as coordenadas de latitude e longitude das geometrias. Por outro lado, os usuários ficaram acostumados a utilizar funcionalidades mais ágeis do MAF, como medir distâncias e realizar consultas espaciais que envolviam o desenho de geometrias no mapa e a captura das suas coordenadas. Nesse sentido, os usuários passaram a solicitar um cadastro de localizações mais prático em que eles apontassem a coordenada no mapa do MAF e desenhassem áreas e linhas para salvá-las como localizações dos acompanhamentos. Entretanto, não se pretendia implementar estas funcionalidades no MGD, de forma que a integridade das obrigações de cada módulo fosse mantida segregada: o MGD com a tarefa de cadastrar e o MAF com a responsabilidade de apresentar. Além disso, a disponibilização de um mapa no MGD despenderia muito tempo de desenvolvimento. Seria necessário replicar uma grande quantidade do código em PHP e *Javascript* do MAF para Java e Ext JS, incluindo o controle de navegação do mapa; e configurar e utilizar a API do MapServer com o Java que oferece muito menos recursos que a API do PHP.

Para contornar estas dificuldades, foi empregada uma extensão da biblioteca do Ext JS chamada Geo Ext [24], cujos componentes foram desenvolvidos para se comunicar com outra biblioteca *open source* em *Javascript* chamada OpenLayers [25]. O OpenLayers é dedicado à apresentação de mapas em um browser independentemente da linguagem utilizada do lado do servidor, já possuindo os controles de navegação no mapa de *zoom* e *pan* embutidos. Com ele, o mapa é renderizado utilizando os recursos de *tiles* para aperfeiçoar seu carregamento e sua movimentação. Não é necessário programar a captura de eventos de clique do *mouse*; estas informações são obtidas a partir de seus objetos. Ele também possui os recursos para desenhar pontos, linhas e áreas poligonais e circulares no mapa, não sendo necessário utilizar as trabalhosas bibliotecas em *Javascript* para tal fim. Utilizando o GeoExt, o mapa do OpenLayers é inserido em uma janela do ExtJS, possibilitando a captura dos eventos de desenho de geometrias para preencher os campos de coordenadas do formulário de cadastro de localização dos acompanhamentos.

O mapa do OpenLayers também é composto por um conjunto de camadas sobrepostas assim como no MapServer. Entretanto, suas camadas podem possuir uma identificação da camada base que funciona como um plano de fundo para o mapa, caso não haja internet disponível. Isto é importante porque o SIPLOM está disponibilizado em uma *intranet* onde não é necessário que haja comunicação com a *internet*.

Normalmente, em outros sistemas, a camada base se refere a algum serviço WMS de mapas comumente utilizado em sistemas *webgis*, como mapas do Google, Open Street Map ou OSGeo. Estes mapas são acessados através da *internet*. Neste caso, se não houver acesso à *internet*, o mapa fica sem o plano de fundo. No caso do SIPLOM, as camadas apontam para o mapa que o usuário já via no MAF. Esta solução pode ser observada na Fig. 1. Para isto, duas ações são fundamentais: a configuração do MapServer para fornecer serviço WMS; e a configuração das camadas do mapa do usuário para estarem disponíveis para tal serviço.

VI. OBSERVAÇÕES FINAIS

O Sistema de Planejamento Operacional Militar (SIPLOM) representa o componente de *software* de Comando e Controle do MD. Ele integra o Sistema Militar de Comando e Controle (SISMC2) e é empregado para a obtenção de uma Consciência Situacional do Teatro de Operações que auxilie os decisores do nível político-estratégico nos seus processos de tomada de decisão. Ele possui uma arquitetura modular que permite o cadastro e o acompanhamento das Forças através de uma visualização gráfica que exibe o posicionamento e os inter-relacionamentos dos meios aéreos, navais e terrestres em um mapa.

O SIPLOM 3 representa um ponto de inflexão no desenvolvimento de *software* de C2 para Defesa, ao aprofundar marcadamente três tendências já presentes nas versões anteriores: simplificação da arquitetura, por meio da redução do número de módulos; extensão do modelo de dados, através da adesão ao modelo JC3IEDM; e a produção de interfaces mais amigáveis para os usuários, possibilitada pela evolução da linguagem *Javascript*. Além disso, a centralização do banco de dados permite uma gerência sobre os dados mais simples e com economia de recursos computacionais, otimizando os parâmetros de operação do sistema.

Do ponto de vista da tecnologia *stricto sensu*, o grande desafio superado foi a compatibilização do MGD, escrito em liguagem Java, e do MAF, escrito em linguagem PHP. A configuração do servidor Apache utilizando o protocolo AJP foi fundamental para redirecionar as requisições servlet/JSP para o Apache Tomcat ou enviá-las para o interpretador do PHP. Desta maneira, conseguimos aproveitar os potenciais de cada linguagem para o projeto SIPLOM: o MAF permaneceu desenvolvido em PHP utilizando a API Mapscript para gerar os mapas pelo MapServer; e o novo MGD pôde ser feito em Java para utilizarmos seus *frameworks* de mapeamento objeto-relacional que agilizam o desenvolvimento do sistema, suprimindo a necessidade de escrevermos incontáveis comandos SQLs para persistirmos e consultarmos as informações no volumoso banco de dados que teve sua complexidade ampliada consideravelmente nesta versão do sistema. É importante ressaltar que as requisições AJAX também contribuíram para a comunicação entre os módulos permitindo a adoção de uma interface de *login* única e a reutilização de rotinas para processamento ou obtenção de dados.

Para futuras versões, vislumbra-se a orientação da arquitetura do SIPLOM para serviço no curto prazo, e o desenvolvimento de um módulo específico de ferramentas de apoio à decisão no longo prazo.

REFERÊNCIAS

[1] Estratégia Nacional de Defesa. Disponível em: <http://www.defesa.gov.br/projetosweb/estrategia/arquivos/estrategia_defesa_nacional_portugues.pdf>. Acesso em 22/06/2012.

[2] Doutrina Militar de Defesa. Disponível em: <https://www.defesa.gov.br/index.php/doutrina-militar.htm>. Acesso em 22/06/2012.

[3] Mulitlateral.Interoperability Programme (MIP). Disponível em: <https://mipsite.lsec.dnd.ca/Public%20Document%20Library/04-Baseline_3.1/Interface-Specification/JC3IEDM/JC3IEDM-Overview-3.1.4.pdf>. Acesso em: 22/06/2012.

[4] M. Corsino, P. Taranti, L. Ouriques, A. Balbi, "MAFWeb: O SIG Defesa", XI Simpósio de Aplicações Operacionais em Áreas de Defesa, São José dos Campos, São Paulo, Setembro de 2009.

[5] PostgreSQL. Disponível em: http://www.postgresql.org.

[6] Regina O. Obe, Leo S. Hsu, "PostGIS In Action", Manning Publication Co., April 2011.

[7] Apache HTTP Server Project. Disponível em: <http://httpd.apache.org/>. Acesso em: 265/06/2012.

[8] Apache Tomcat. Disponível em: <http://tomcat.apache.org>. Acesso em: 265/06/2012.

[9] R. T. Fielding, 2000, "Architectural Styles and the Design of Network-based Software Architectures", Doctor Philosophy in Doctor of Philosophy in Information and Computer Science, University of California, Irvine.

[10] C. Bauver, G. King, "Java Persistence with Hibernate", Manning Publication Co., 2007.

[11] M. Keith, M. Schnicariol. "Pro JPA 2 Mastering the Java™ Persistence API", Apress, 2009.

[12] P. Dall`Oglio "PHP Programando com Orientação a Objetos", Novatec, 2007.

[13] MapServer. Disponível em: <http://www.mapserver.org>/, Acesso em: 265/06/2012.

[14] M. Vasconcelos, M. Guidini, "O Uso de Padrão OpenGIS em Sistemas Militares de Comando e Controle", Instituto Tecnológico de Aeronáutica (ITA), São José dos Campos, São Paulo.

[15] Open Geospatial Consortium. Disponível em: http://www.opengeospatial.org/. Acesso em: 22/06/2012

[16] Glassfish – Open Source Application Server. Disponível em: <http://glassfish.java.net/>. Acesso em: 265/06/2012.

[17] What is the Document Object Model?, November 2000. Disponível em: <http://www.w3.org/TR/DOM-Level-2-Core/introduction.html>. Acesso em: 22/06/2012.

[18] World Wide Web Consortium (W3C). Disponível em: <http://www.w3.org/. Acesso em: 22/06/2012.

[19] JSON. Disponível em: <http://www.json.org>. Acesso em: 22/06/2012

[20] JQuery Javscript Library: Disponível em: <jquery.com>. Acesso em: 22/06/2012

[21] Prototype Javascript Framework. Disponível em: <http://www.prototypejs.org>. Acesso em: 22/06/2012

[22] S. Frederick, C. Ramsay, S. Blades, N. White, "Learning Ext JS 3.2", Packt Publishing Ltd., October 2010

[23] N. Harrison, B. Foote, H. Rohnert. "Pattern Languages of Program Design 4". Software Pattern Series. Addison-Wesley, 1999.

[24] Javascript Toolkit for Tich Web Mapping Applications - GeoExt. Disponível em: <http://www.geoext.org>. Acesso em: 23/06/2012

[25] E. Hazzard "OpenLayers 2.10: Beginner`s Guide", Packt Publishing Ltd., March 2011