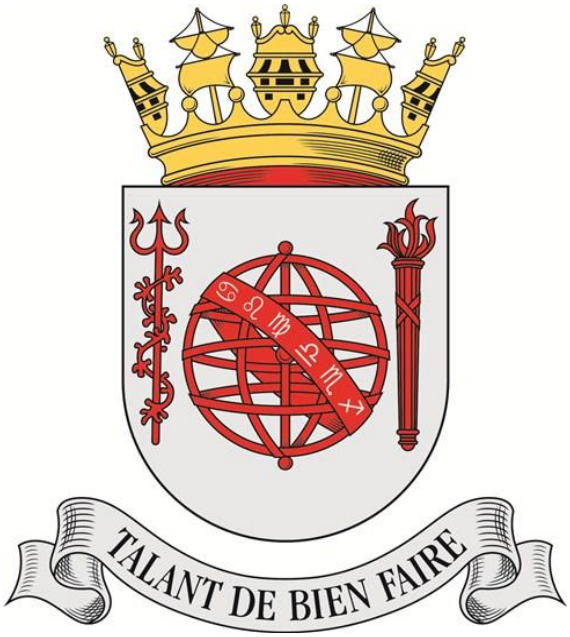
ESCOLA NAVAL

DEPARTAMENTO DE CIÊNCIAS DO MAR



Módulo de alertas com base em dados AIS para apoio à vigilância marítima

Pedro Miguel Caroço Fernandes

MESTRADO EM CIÊNCIAS MILITARES NAVAIS

(MARINHA)

2014

ESCOLA NAVAL

DEPARTAMENTO DE CIÊNCIAS DO MAR

DISSERTAÇÃO DE MESTRADO EM CIÊNCIAS MILITARES NAVAIS

Módulo de alertas com base em dados AIS para apoio à vigilância marítima

O Mestrando,	O Orientador,	O Coorientador
(assinado no original)	***(assinado no original)***	***(assinado no original)***
ASPOF Caroço Fernandes	CFR Cavaleiro Ângelo	1TEN Gonçalves de Deus

Agradecimentos

Em primeiro lugar, quero agradecer ao meu orientador, o CFR Cavaleiro Ângelo, e ao meu coorientador, o 1TEN Gonçalves de Deus, por todo o incentivo e disponibilidade demonstrada durante realização da presente dissertação de mestrado e sem os quais a realização da mesma não teria sido possível.

Expresso também os meus sinceros agradecimentos à DAGI e CADOP, em especial ao 1TEN Rodrigues António pelo contributo prestado na realização deste trabalho.

Agradeço também a todas as pessoas que contribuíram para minha formação ao longo da Escola Naval e que fizeram de mim a pessoa e militar que sou hoje.

Aos Índios, por todos os momentos que passamos durante estes últimos quatro anos.

Por fim, quero agradecer à minha família por todo o apoio e paciência ao longo destes anos.

Dedicatória

À minha família, que fez com que este "navio" chegasse a bom porto.

Resumo

A Marinha é responsável pela vigilância e fiscalização dos espaços marítimos sob soberania e jurisdição nacional o que exige a compilação e análise de uma grande quantidade de dados. Neste sentido, o sistema AIS e os dados por ele fornecidos desempenham um papel preponderante para o desenvolvimento do Conhecimento Situacional Marítimo (CSM) nas águas sob jurisdição portuguesa. A informação fornecida por este sistema permite identificar e seguir os movimentos dos navios, possibilitando a criação de conhecimento até então praticamente inexistente. Neste âmbito, têm surgido nos últimos anos várias aplicações para gerir e monitorizar o panorama marítimo de superfície, numa perspetiva de apoio às operações correntes, mas também numa perspetiva de análise de informações e *Business Intelligence* (BI), como é o caso do protótipo AISINTEL. Este protótipo, desenvolvido pela Marinha, visa melhorar o CSM e permitir o teste e experimentação de novas funcionalidades de BI a partir de dados georreferenciados da navegação (dados AIS e MONICAP) e de outras fontes de dados relacionadas com o ambiente marítimo (dados METOC). Esta ferramenta teve a sua origem em dissertações de mestrado de alunos da Escola Naval na área do CSM, e tem conhecido um desenvolvimento incremental baseado em módulos independentes mas interligados entre si.

A presente dissertação pretende dar continuidade aos trabalhos desenvolvidos nesta área, contribuindo de forma concreta para o desenvolvimento de um novo módulo que permita o aviso antecipado de situações de risco, visando a diminuição da ocorrência de acidentes marítimos e monitorização de ocorrências que denotem algum comportamento suspeito.

Palavras chave: Conhecimento Situacional Marítimo, AISINTEL, Alertas AIS, Sistema AIS.

Abstract

The Portuguese Navy is responsible for monitoring and surveillance of the maritime areas under national jurisdiction and sovereignty which requires the compilation and analysis of a large amount of data. The AIS system and its data play an important role for the development of a Maritime Situational Knowledge (MSK) capability in the Portuguese areas of jurisdiction. The information provided by this system allows the identification and tracking the ships movement, enabling the creation of knowledge hitherto non-existent. In this context, several applications have emerged in recent years to manage and monitor the surface picture in order to support current operations and perform intelligence analysis and Business Intelligence (BI), as is the case of the prototype AISINTEL. This prototype, developed by the Portuguese Navy, aims to improve MSK and allows the test and experimentation of new BI functionalities from geo-referenced navigation data (AIS and MONICAP data) and other sources of data related to the marine environment (METOC data). This tool has its origins on the Naval Academy dissertations in the area of MSK and has known an incremental development based on independent but interconnected modules.

This dissertation intends to carry on the work already made in this area, contributing thoroughly to the development of a new module that allows early warning of hazardous conditions, aiming to reduce the occurrence of maritime accidents and monitoring occurrences denoting any suspicious behaviour.

Keywords: Maritime Situational Knowledge, AISINTEL, AIS alarms, AIS system.

Epígrafe

"Nenhum de nós poderá, num momento qualquer, garantir que a sua doutrina seja a que encerre verdade (...)"

Agostinho Silva

Índice

Lista de figuras

Lista de Tabelas

Lista de Gráficos

Entrevistas

GDH: 21/11/2012

Local: CADOP

Intervenientes: CTEN Cavaleiro Ângelo, 1TEN Rodrigues António, 1TEN Gonçalves de Deus, CAD Caroço Fernandes

Assunto: Alarmística com base em dados AIS, identificação de requisitos funcionais para a criação e monitorização de alarmes com base em dados AIS

GDH: 21/11/2013

Local: Esquadrilha de Submarinos

Intervenientes: CTEN Pereira de Sousa, 1TEN Guerreiro, 1TEN Gonçalves de Deus, ASPOF Caroço Fernandes

Assunto: Funcionalidades de *Intelligence* do TransViewer 32

Lista de siglas e acrónimos

1TEN	Primeiro-tenente
AIS	*Automatic Identification System*
AOM	Área Operacional da Marinha
ASPOF	Aspirante a official
ATA	*Actual Time of Arrival*
ATD	*Actual Time of Departure*
ATP	*Allied Tactical Publication*
C2	Comando e Controlo
C3	*Command, Control and Communications*
CAD	Cadete
CADOP	Centro de Gestão e Análise de Dados Operacionais
CCIS	Command and Control Information System
CCMAR	*Allied Maritime Component Command*
CEMA	Chefe do Estado-Maior da Armada
CFR	Capitão-de-fragata
CINAV	Centro de Investigação Naval
CMG	Capitão-de-mar-e-guerra
CNUDM	Convenção das Nações Unidas sobre o Direito do Mar
COI	*Contact Of Interest*
COMAR	Centro de Operações Marítimas
COMNAV	Comando Naval
CPA	*Closest Point of Approach*
CSM	Conhecimento Situacional Marítimo
CTEN	Capitão-tenente
CZMS	Comando da Zona Marítima do Sul
DAGI	Direção de Análise e Gestão da Informação
DGAM	Direção Geral da Autoridade Marítima
DITIC	Direção das Tecnologias de Informação e Comunicação
DRISUB	Esquadrilha de Submarinos

DSC	*Digital Selective Call*
EMSA	*European Maritime Safety Agency*
ETA	*Estimated Time of Arrival*
FEDER	Fundo Europeu de Desenvolvimento Regional
FEUP	Faculdade de Engenharia da Universidade do Porto
FTP	*File Transfer Protocol*
GDH	Grupo Data Hora
GPE-SV	Grupo de Planeamento Estratégico para o Sistema de Vigilância da Marinha
GPIAM	Gabinete de Prevenção e de Investigação de Acidentes Marítimos
GPS	*Global Positioning System*
GRIB	*General Regularly-distributed Information in Binary form*
IMDatE	*Integrated Maritime Data Environment*
IO	Investigação Operacional
JMCIS	*Joint Maritime Command Information System*
KML	*Keyhole Markup Language*
MARINTSUM	*Maritime Intelligence Summary*
MATLAB	*MATrix LABoratory*
MCCIS	*Maritime Command and Control Information System*
MMSI	*Maritime Mobile Service Identity*
MONICAP	Monitorização Contínua das Atividades da Pesca
MRCC	*Maritime Rescue Coordination Centre*
MRS	*Mandatory ship Reporting System*
MRSC	*Maritime Rescue Subcentre*
MSK	*Maritime Situational Knowledge*
NATO	*North Atlantic Treaty Organization*
NAVSITREP	*Navy Situational Report*
NM	*Nautical Mile*
NRP	Navio da República Portuguesa
PIMTRACK	*Position and Intended Movement Track*
QREN	Quadro de Referência Estratégico Nacional
RAM	*Random Access Memory*
RAP	*Recognized Air Picture*

RIEAM	Regulamento Internacional para Evitar Abalroamentos no Mar
RLP	*Recognized Land Picture*
RMP	*Recognized Maritime Picture*
SACLANT	*Supreme Allied Commander Atlantic*
SADAP	Sistema de Apoio à Decisão para a Atividade de Patrulha
SAR	*Search and Rescue*
SIG	Sistema de Informação Geográfica
SPAWAR	*Space and Naval Warfare Systems Command*
SRR	*Search and Rescue Region*
SSTI	Superintendência dos Serviços de Tecnologias da Informação
SWH	*Significant Wave Height*
TCPA	*Time to Closest Point of Approach*
TV32	*Transview*
TWD	*True Wind Direction*
TWS	*True Wind Speed*
UNODC	*United Nations Office on Drugs and Crime*
VIPS	*Vessel Identification and Positioning System*
VMS	*Vessel Monitoring System*
VTS	*Vessel Traffic Service*
WAN	*Wide Area Network*
WDir	*Wave Direction*
WISE	*Web Information Services Environment*
ZEE	Zona Económica Exclusiva

Capítulo 1

Introdução

1.1 Enquadramento

1.2 Justificação do tema

1.3 Objetivos

1.4 Questões de investigação

1 Capítulo 1 – Introdução

Desde sempre Portugal foi uma Nação virada para o mar, possuindo uma posição geoestratégica invejável, servindo de ponte entre três continentes – Europa, África e América. Apesar de Portugal possuir uma área terrestre de pequenas dimensões, possui uma extensíssima área marítima que corresponde a cerca de 18,7 vezes o território terrestre nacional[1]. Este facto faz com que o Estado Português assuma responsabilidades a nível internacional, nas quais se destaca a busca e salvamento marítimo, *Search and Rescue* (SAR). Atualmente, o Estado Português tem uma área de responsabilidade, no que diz respeito à busca e salvamento marítimo, de 5 792 740 km^2, o que corresponde a cerca de 63 vezes a superfície do território nacional[1]. A vasta área marítima que Portugal possui torna-o, um país com uma pequena porção de território terrestre, mas com uma vasta área marítima sob sua jurisdição.

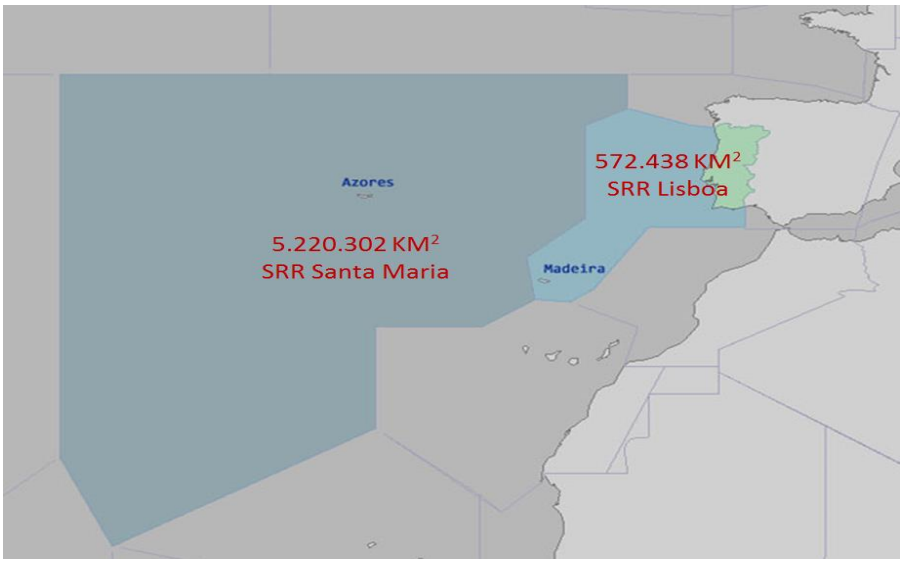


Figura 1 - Áreas de responsabilidade nacional[2].

Atualmente, o tráfego marítimo é responsável por cerca de 90% do comércio mundial, por isso, mais do que nunca, Portugal encontra-se numa posição estratégica. Facto é que o mar assume cada vez mais um maior peso na economia portuguesa, pois nas águas sob jurisdição portuguesas passam cerca de 53% do comércio externo à União Europeia e 60% do comércio externo português[1]. É através do mar que cerca de 70%

[1]MARINHA (s.d.). *Portugal, uma nação marítima*. Retirado de, http://www.marinha.pt/pt-pt/historia-estrategia/estrategia/folhetospt/Portugal_uma_nacao_maritima.pdf, consultado a 3de outubro de 2013.
[2] ESQUADRA 601 (2013). Busca e salvamento [imagem].Retirado de http://www.lobos601.net/sar.html, consultado a 12 de agosto de 2014.

das importações nacionais são feitas, o que inclui a totalidade do petróleo e 2/3 do gás natural consumidos em território português. Para Portugal o mar já representa 5 a 6% do PIB (Produto Interno Bruto), projetando-se que até 2025 este valor cresça para 10 a 12%[3].

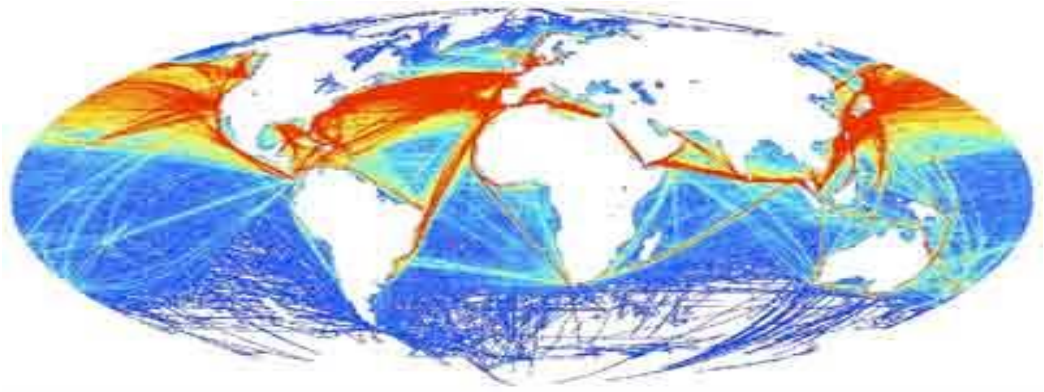


Figura 2 - Densidade do tráfego marítimo a nível mundial[4].

Como podemos constatar o mar assume um papel fundamental, não só na economia portuguesa como mundial. Deste modo, é necessário garantir a segurança de quem o usa. Neste sentido a Marinha tem procurado desenvolver ferramentas que auxiliem e potenciem as capacidades do seu sistema de vigilância marítima, em particular, no desenvolvimento de ferramentas que reforcem a construção do Conhecimento Situacional Marítimo (CSM). Com a criação do GPE-SV[5], foi reconhecida a importância do conceito de CSM e da importância para a Marinha em dispor de um Sistema de Vigilância com capacidade IVDAR[6]. A análise de dados AIS e a construção de conhecimento situacional a partir desta fonte de dados esteve na origem do desenvolvimento da ferramenta AISINTEL.

Uma das ferramentas para construção de CSM, desenvolvidas no seio da Marinha, e já referida anteriormente, é o AISINTEL[7]. Este protótipo, desenvolvido pela DAGI, permite a análise e visualização integrada de dados AIS[8] e MONICAP[9]. Esta ferramenta

[3] De acordo com o estudo realizado pela Sociedade de Avaliação Estratégica e Risco (SaeR), relativamente ao *hypercluster* da economia do mar, 2009.
[4] [s.n.] (2013). *Tráfego marítimo*[imagem].Retirado de http://nomehagasmuchocaso.wordpress.com/tag/triangulo-bermudas/, consultado a 03-05-2014.
[5] GPE-SV – Grupo de Planeamento Estratégico para o Sistema de Vigilância da Marinha – despacho do ALM CEMA nº18/09 de 18 de maio de 2009.
[6] IVDAR = ISTAR – *Intelligence, Surveillance, Targeting, Acquisition and Reconnaissance*.
[7] O AISINTEL é um sistema desenvolvido pela Marinha sob a coordenação do 1TEN Gonçalves de Deus, coorientador desta dissertação, e que será abordado posteriormente na subsecção 2.2.
[8] O AIS é um sistema de identificação automática que transmite as mensagens de dados através de ondas rádio (VHF), com um alcance médio de 25 NM. O alcance deste sistema depende de diversos fatores, tais como: condições meteorológicas, elevação da antena e potência de transmissão.

disponibiliza um conjunto variado de indicadores de CSM (Melo L., 2010; Deus R., 2011), assim como um conjunto diversificado de funcionalidades que apoiam a Marinha no processo de tomada de decisão e produção de *Business Intelligence* (Melo H., 2011). O AISINTEL é constituído por diferentes módulos com especificidades próprias. Um dos últimos módulos a ser desenvolvido foi o módulo de análise de padrões que permite caraterizar áreas marítimas, através de mapas de temperatura relativamente à ocorrência de eventos de interesse (Melo H., 2011). A caraterização destes eventos de interesse presentemente é feita apenas com base num histórico de dados AIS. Posteriormente ao desenvolvimento deste módulo, foi suscitado pelo CADOP a necessidade em dispor de uma capacidade de análise de eventos de interesse em tempo real. A esta capacidade de análise corresponde a disponibilização de um conjunto de alarmes sempre que ocorram determinadas condições previamente definidas pelo decisor. Um exemplo de um evento de interesse é o acompanhamento de contactos de interesse ou entrada em áreas previamente consideradas de interesse.

A presente dissertação vai de encontro aos esforços da Marinha na melhoria da sua capacidade de esclarecimento do panorama situacional marítimo, através do desenvolvimento de um módulo de alertas com base em dados AIS. Espera-se ainda que o módulo desenvolvido seja alvo de apreciação para que seja possível a sua incorporação no protótipo AISINTEL e *Oversee*[10].

Esta dissertação encontra-se dividida em cinco capítulos. No primeiro capítulo será feito o enquadramento do problema em estudo, abordando a importância que ferramentas como AISINTEL têm para um sistema nacional de busca e salvamento eficaz. Também será abordado neste capítulo a justificação do tema, os objetivos que se propõem atingir e as questões de investigação levantadas. No segundo capítulo é feito uma revisão de literatura relativamente a ferramentas desenvolvidas pela Marinha e outras entidades internacionais, que contribuem para a construção de CSM dos espaços sob soberania e jurisdição nacionais. No capítulo seguinte é descrito como foi especificado e implementado o módulo de alertas, definindo a estrutura de um alerta e caracterizando as componentes que a integram. No quarto capítulo são discutidos os

[9] O MONICAP é o sistema de monitorização contínua das atividades da pesca, que utiliza o sistema GPS para localizar os navios e o sistema *Inmarsat C* para estabelecer comunicações satélite entre os navios e os centro de controlo costeiro.

[10] *Oversee* é um sistema que surge no âmbito do desenvolvimento do projeto *Blue Eye*, encontrando-se a ser desenvolvimento pela empresa *Critical Software* em parceria com a Marinha e que será abordado posteriormente neste trabalho na subsecção 2.1.7.

resultados obtidos. No quinto e último capítulo são retiradas algumas ilações sobre o estudo realizado e apresentadas recomendações para trabalhos futuros.

1.1 Enquadramento

Portugal é um país com vasta área marítima, onde vêm confluir as mais importantes e movimentadas rotas marítimas internacionais. Desta forma, cabe à Marinha garantir a segurança e autoridade do Estado no mar, facto confirmado na Lei Orgânica da Marinha (Decreto-Lei nº233/2009 artº2 nº3 alínea a) e b)).

Figura 3 - Funções e tarefas da Marinha[11].

Neste âmbito, a Marinha desempenha as funções típicas de uma Armada e de uma Guarda Costeira, o que permite uma eficiente gestão dos recursos públicos, que de outro modo, exigiria duas estruturas diversificadas e potencialmente redundantes.

Neste contexto, a Marinha é responsável pela vigilância e fiscalização de uma extensíssima área marítima, apesar de se encontrar atualmente fortemente limitada quer em termos de pessoal, quer em termos de meios. O corte orçamental efetuado nos últimos anos, veio diminuir o treino dos militares, limitar as taxas de navegação das unidades navais, assim como a manutenção das mesmas, levando a Marinha "*a navegar*

[11] MARINHA (2011). *Diretiva de Política Naval 2011*[imagem]. Lisboa, Marinha.

e operar no limite"[12]. No entanto, apesar das fortes restrições orçamentais, a qualidade do desempenho operacional da Marinha, pode ser verificado na elevada taxa de sucesso anual do serviço de busca e salvamento marítimo, que situa-se em média entre os 96% e 98%, distinguindo-se como um valor de referência a nível mundial.

Desta forma, é fundamental o desenvolvimento de competências técnicas dirigidas para o combate à criminalidade por via marítima[13] e à salvaguarda da vida humana no mar. Hoje em dia, devido aos inúmeros equipamentos existentes a bordo dos navios, chegam aos MRCC's uma grande quantidade de dados. Neste sentido, e de modo a tirar o máximo proveito destes, a Superintendência dos Serviços de Tecnologias da Informação (SSTI), na sua diretiva sectorial, definiu como um dos objetivos "*contribuir para o desenvolvimento de uma capacidade de construção do conhecimento situacional marítimo*"[14], que se tem materializado no acompanhamento e coordenação de projetos de IID[15] na área do CSM, do qual o projeto Blue Eye é um exemplo, assim como, no desenvolvimento de protótipos, também nesta área, onde se insere o AISINTEL.

Neste âmbito, surgiu, em 2010, o protótipo AISINTEL, que utiliza os dados fornecidos pelo sistema AIS e MONICAP, com o objetivo de disponibilizar um conjunto de indicadores estatísticos relativos ao panorama marítimo. Desde 2010, até ao presente momento, foram integrados diversos conjuntos de funcionalidades que apoiam o processo de *Intelligence* para teste e experimentação dos mesmos. Atualmente, esta ferramenta possui diversos módulos que permitem ao utilizador efetuar o seguimento de vários navios, ver a proximidade entre navios e outras funções que serão descritas mais à frente na secção 2.2.

Paralelamente ao desenvolvimento do protótipo AISINTEL, foram desenvolvidas outras aplicações que permitem o contínuo registo, descodificação e compilação de dados AIS, que na sua essência sustentam todas as funcionalidades do AISINTEL. Toda a análise que o AISINTEL permite requer um rápido acesso a um histórico de dados AIS. Este histórico está em permanente construção desde janeiro de 2010. Sendo necessário garantir a criação de um histórico de dados que permitisse reconstruir o panorama marítimo em qualquer dia de um ano, foi necessário desenvolver um conjunto

[12] SALDANHA LOPES, Almirante (2013, 26 de maio). *Discurso do Chefe do Estado-Maior da Armada*. Retirado de, http://www.marinha.pt/pt-pt/media-center/noticias-destaques/Paginas/Discurso-Chefe-Estado-Maior-Armada.aspx, consultado a 5 de fevereiro de 2014.
[13] De acordo com o último relatório da UNODC (*United Nations Office on Drugs and Crime*) Portugal é um dos principais destinos das organizações criminosas da América do Sul e países lusófonos.
[14] MARINHA (2011). *Directiva Sectorial da Superintendência dos Serviços de Tecnologias da Informação 2011*, Lisboa, Marinha, pp13.
[15] IID – Investigação, Inovação e Desenvolvimento.

de aplicações que efetuasse o registo diário e ininterrupto das mensagens AIS, a sua descodificação e compilação num formato que pudesse ser trabalho por esta ferramenta. Desta forma foram desenvolvidas em MATLAB três aplicações para efetuar o registo, descodificação e compilação de dados AIS. Por "registo" entende-se a leitura de mensagens AIS no formato NMEA[16] e seu registo usando num ficheiro ".mat" através de uma estrutura de dados designada por *cell array*[17]. Esta estrutura de dados associa a mensagem AIS com o GDH em que foi registada. A "descodificação" das mensagens AIS consiste em retirar informação relativa ao navio que enviou a mensagem de forma a obter a respetiva posição, velocidade e rumo, entre outros atributos possíveis de descodificar. A "compilação" consiste em relacionar os atributos descodificados numa tabela que possa ser carregada pelo AISINTEL e sujeita à aplicação de filtros consoante as necessidades de análise. O sistema que garante o registo, descodificação e compilação de dados AIS é, à data, constituído por quatro servidores, três aplicações distintas, e um cliente que acede aos dados por FTP (*File Transfer Protocol*). Na figura abaixo estão descritos os elementos que compõem este sistema.

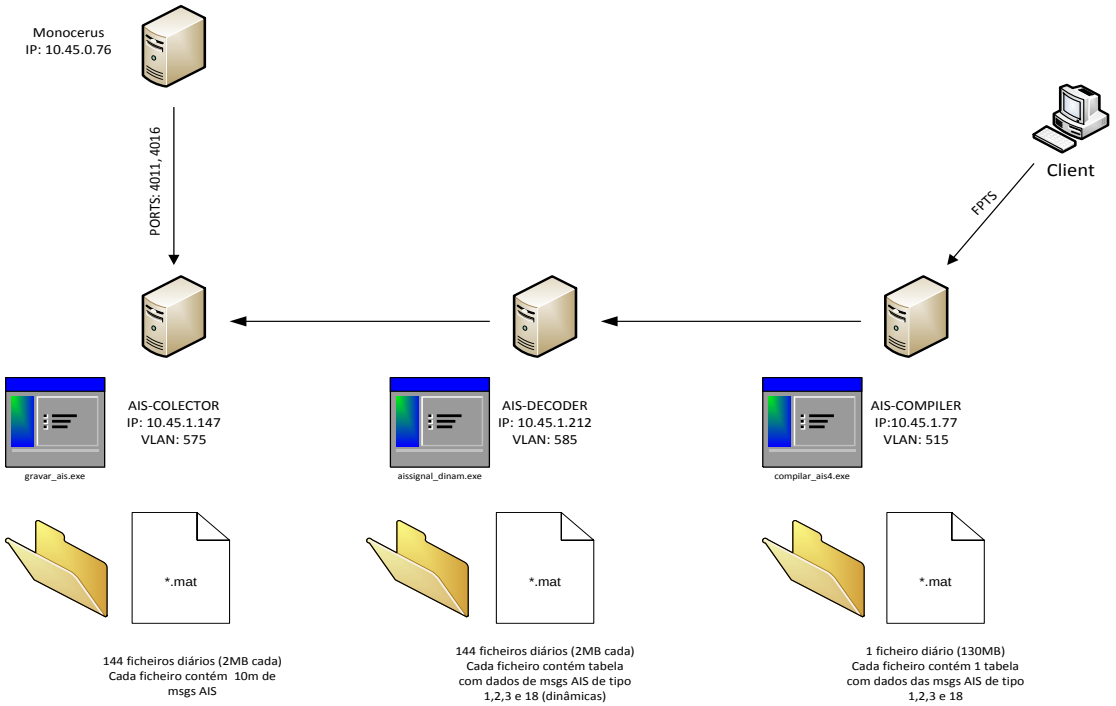


Figura 4 - Arquitetura do sistema para registo, descodificação e compilação de dados AIS.

[16] Exemplo de mensagem AIS no formato NMEA:
!ABVDM,1,1,,A,13s<kKUP0twHnrbE5nJ`sgwF28Ig,0*75

[17] Um *cell array* é um conjunto de várias células, em que cada uma delas pode conter qualquer tipo de dados, como: matrizes, texto, número e combinações de texto e números, entre outros.

As três aplicações que realizam as tarefas de registo, descodificação e compilação funcionam 24 horas por dia, 365 dias por ano. Na figura abaixo está representada a primeira aplicação do sistema de registo de mensagens AIS (AIS-COLLECTOR):

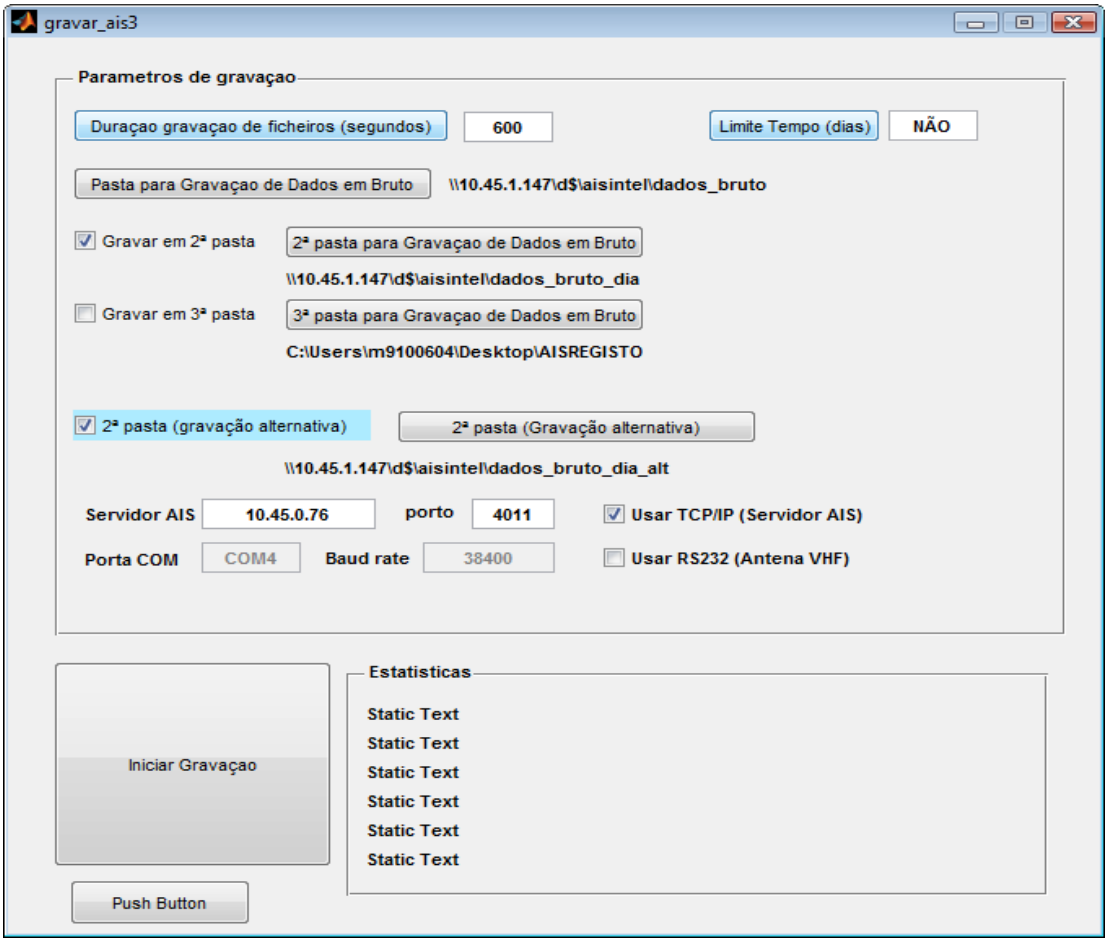


Figura 5 - Aplicação que efetua o registo contínuo de MSG AIS.

O sistema referido anteriormente disponibiliza, quase em tempo real[18], informação de navios equipados com o *transponder* AIS ao protótipo AISINTEL. Em particular, a aplicação "gravar_ais3" efetua a construção dos ficheiros diários que contêm as posições AIS após a descodificação das respetivas mensagens AIS de tipo 1, 2, 3 e 18. Estes ficheiros diários são utilizados pelo módulo de alertas e permitem a monitorização, quase em tempo real, de eventos de interesse.

Apesar de este protótipo contemplar um variado leque de funcionalidades que apoiam o processo de *Intelligence*, verificou-se a necessidade de desenvolver um módulo de alarmística que permita ao utilizador total flexibilidade na definição e parametrização daquilo que pretende ser alertado. Por conseguinte, tornou-se necessário

[18] O registo de mensagens ocorre num período de 10 minutos. A descodificação de um ficheiro de 10 minutos demora cerca de 5 minutos. A integração das mensagens AIS na tabela diária demora sensivelmente 15 minutos. Assim, entre a chegada de uma mensagem AIS ao servidor MONOCERUS e a sua disponibilização no AISINTEL decorrerá um período de aproximadamente 15 minutos.

analisar o conceito de "alerta" no sentido de identificar as diferentes estruturas de dados ou dimensões que a integram, para que a implementação do alerta consiga cobrir o maior número de situações de interesse suscitadas pelo utilizador.

1.2 Justificação do tema

A missão da Marinha abrange diversas tarefas, onde se destaca a Segurança e Autoridade do Estado. No presente enquadramento legal, a Marinha é responsável por "*exercer a autoridade marítima e garantir o cumprimento da lei nos espaços marítimos sob soberania ou jurisdição nacional*"[19], assim como é responsável por "*assegurar o serviço de busca e salvamento marítimo nos espaços marítimos sob responsabilidade nacional*"[20]. De forma a cumprir as tarefas acima enunciadas é imperativo que a Marinha disponha e utilize de forma eficaz e eficiente os diversos recursos que tem à sua disposição. Neste sentido, as ferramentas de apoio à decisão desempenham um papel crucial na tomada de decisão que conduzem a um empenhamento mais adequado e eficaz de diversos meios, em particular, dos meios navais.

A presente dissertação de mestrado tem como objetivo o desenvolvimento de uma aplicação informática, designada por "módulo de alertas", que permita o aviso antecipado de situações de risco, procurando a diminuição da ocorrência de acidentes marítimos e deteção de padrões suspeitos. Desta forma, esta dissertação constitui um desafio, pois terá implicações imediatas ao nível operacional, constituindo uma oportunidade de beneficiar os projetos em curso nesta área. O presente estudo pretende também dar continuidade ao trabalho desenvolvido com o protótipo AISINTEL, procurando desenvolver novas funcionalidades que permitam tirar o melhor proveito dos dados ao dispor da Marinha, contribuindo para um melhor esclarecimento do CSM. Este trabalho pretende assim, através do desenvolvimento de uma aplicação informática, contribuir de forma ativa para o cumprimento do objetivo sectorial, definido na Diretiva Sectorial da SSTI, sendo que este objetivo contribui para a concretização do objetivo estratégico nº6 "*Consolidar e potenciar a capacidade de Conhecimento Situacional Marítimo (CSM)*"[21], vertida na Diretiva de Planeamento Naval 2014. No plano académico, pretende-se alicerçar conhecimentos na área dos sistemas de informação, em

[19] De acordo com a alínea a) do nº3 do art.º2 do Decreto-Lei nº233/2009, de 15 de setembro (Lei Orgânica da Marinha).
[20] De acordo com a alínea b) do nº3 do art.º2 do Decreto-Lei nº233/2009, de 15 de setembro (Lei Orgânica da Marinha).
[21] MARINHA (2014). *Diretiva de Planeamento Naval 2014*. Lisboa, Marinha, pp9.

particular, na construção de alertas a partir de dados georreferenciados e na sua avaliação e adequabilidade operacional.

1.3 Objetivos

A presente dissertação de mestrado tem como objetivo desenvolver uma aplicação informática, em MATLAB, utilizando dados AIS, de forma a avaliar automaticamente situações de alerta pré-definidas pelo utilizador. Para isso foram estabelecidos os seguintes objetivos:

1. Implementação de rotinas para a construção de alarmes definidos pelo utilizador, tendo em consideração diversos fatores, tais como: área, janela temporal, condições meteo-oceanográficas, listas de navios e dados do alerta;
2. Implementação de rotinas para monitorização dos alertas;
3. Estudo dos algoritmos implementados para deteção de alarmes através da elaboração de estatísticas.

1.4 Questões de investigação

Nesta dissertação pretende-se abordar diversas questões, sendo as mais relevantes as que estão relacionadas com a realidade operacional da Marinha. Para tal foram definidas as seguintes questões:

1. Como desenhar a arquitetura de informação de um alerta e como implementá-la usando a linguagem MATLAB?
2. A arquitetura proposta para um alarme é adequada tendo em consideração o propósito para o qual o alarme foi criado? Como medir esta adequabilidade?
3. Face a um conjunto pré-definido de situações de interesse na área SAR, qual a sua distribuição geográfica e temporal nas SRR de Lisboa e Santa Maria?

Durante a realização deste trabalho poderão ser levantadas outras questões de caráter computacional e de modelação matemática dos alarmes, como por exemplo questões relacionadas com a eficiência dos algoritmos utilizados para pesquisar navios que satisfaçam as condições lógicas inerentes ao alarme em causa.

Capítulo 2

Revisão da Literatura

2.1 Sistemas de informação na área do CSM

2.2 AISINTEL e trabalho precedente

2.3 TRITON – futuro sistema de CSM NATO

2.4 Métricas para desenvolvimento de *software*

2 Capitulo 2 – Revisão da Literatura

Os sistemas de informações na área do CSM são, hoje em dia, uma mais-valia para as entidades responsáveis pelo controlo da navegação nas áreas portuárias e nas zonas costeiras em todo o mundo. Estes sistemas fornecem a posição geográfica de um navio, em tempo real, permitindo às autoridades identificarem potenciais casos de perigo, de modo a tomarem as medidas adequadas de forma atempada.

Neste capítulo pretende-se definir, sucintamente, o que é um sistema de informação na área do CSM e descrever, por ordem cronológica, os principais sistemas de informação de CSM existentes e em uso na Marinha Portuguesa, assim como o futuro sistema TRITON, ainda em desenvolvimento pela NATO.

Nesta área é de realçar os esforços que a Marinha tem vindo a desenvolver no sentido de dotar as suas unidades com sistemas na área do CSM, de forma a conseguir um melhor esclarecimento do panorama de superfície. Em setembro de 2011 foi estabelecido um protocolo com a empresa *Critical Software* para o desenvolvimento de um sistema do tipo C2 (Comando e Controlo), designado *Oversee*[22]. É também de realçar o trabalho desenvolvido pelo CADOP, DAGI e Escola Naval no desenvolvimento do protótipo AISINTEL, que se encontra atualmente em fase de testes e experimentação. Esse protótipo tem servido como um laboratório de ideias e experimentação de vários requisitos técnicos e operacionais na área do CSM, que posteriormente são migrados para sistemas de informação como é o caso do sistema *Oversee*.

Ainda neste capítulo são referidos tipos de métricas para avaliação de *software* no sentido de enquadrar o leitor com as métricas que serão apresentadas no capítulo 4.

2.1 Sistemas de informações na área do CSM

Os sistemas de informação na área do CSM têm, ao longo dos anos, sido objeto de estudo e debate, uma vez que a informação que estes fornecem, contribui de forma decisiva para a tomada de decisões e emprego de meios. Desta forma é importante definir o conceito de CSM. Segundo S.Exª. o Contra-almirante Gameiro Marques o

[22] Em 8 de setembro de 2011 foi estabelecido um protocolo de colaboração entre a Critical Software e a Marinha para o projeto "Sistema de suporte às operações marítimas", designado por Blue Eye.

CSM é "*a criação de saber acerca do espaço marítimo de ação ou de envolvimento (de interesse nacional ou conjuntural), com o objetivo de prever, identificar e localizar situações de interesse e propiciar a tomada de decisões atempadas e mais informadas, que levem a que as ações subsequentes produzam os efeitos desejados no tempo e na medida dos interesses de quem as toma*"[23] (Marques, A., 2013).

Atualmente é cada vez mais importante conhecer o que se passa no mar, de forma a garantir a proteção e segurança de quem o utiliza, mitigando as atividades ilegais e garantindo a livre circulação de pessoas e bens, assim como o cumprimento da CNUDM[24]. Para se efetivar um contributo válido neste campo, é necessário ter conhecimento dos eventos que aí ocorrem. O conhecimento sobre a área de responsabilidade nacional é vital para que seja possível, entre outros fins, identificar e localizar ameaças e riscos, potenciais ou reais, de forma a permitir uma tomada de decisão e emprego de meios atempadamente.

É neste âmbito que os sistemas de informação de CSM desempenham um papel crucial, pois disponibilizam aos seus utilizadores diversas ferramentas que permitem o seguimento de navios, medição de distâncias, desenho de áreas, definição de alertas, entre outras funções para o apoio à tomada de decisão. Estas ferramentas tornam o mar mais seguro, promovendo o desenvolvimento de diversas atividades económicas por esta via.

Segundo Marques (2013), a construção de CSM segue um esquema muito semelhante ao ciclo OODA[25], onde os sistemas de informação assumem um papel muito preponderante nas fases de observação e orientação. Em particular, na fase de orientação, onde a informação recolhida é analisada, avaliada e interpretada, ocorre a produção de conhecimento que constitui o elemento fundamental para a tomada de decisão. No ciclo da construção de CSM é necessário percorrer várias fases, sendo a primeira a fase de vigilância, que consiste na recolha de dados da área a monitorizar. A informação é posteriormente convertido em *intelligence* através do correlacionamento, avaliação, análise, integração e interpretação da informação. A isto corresponde a fase de processamento. Após o tratamento da informação entra-se na fase da disseminação

[23] GAMEIRO MARQUES, Contra-almirante (2013). *Cibersegurança e conhecimento situacional marítimo*, in Revista da Armada, Março 2013, 2013,pp.13.
[24] Convenção das Nações unidas sobre o Direito do Mar – De uma forma geral esta convenção define os direitos e as responsabilidades dos Estados no que diz respeito ao mar e estabelece as fronteiras marítimas.
[25] O ciclo OODA refere-se ao ciclo de decisão de observar, orientar, decidir e agir, desenvolvido pelo estrategista militar Coronel John Boyd.

da mesma, onde as informações são, de forma adequada e por meios apropriados, difundidas para aqueles que delas necessitam. A fase seguinte é a fase de decisão, em que entra a dimensão humana e dá-se o processo decisório. A partir deste ponto estar-se-á em condições de entrar na fase de ação, onde são empenhados os meios de forma coordenada face ao propósito em causa. Deve-se, no entanto, ter em conta que os meios empenhados poderão interagir com a área de operações modificando-a e reiniciando-se o ciclo.

Figura 6 - Ciclo da construção de CSM.

É evidente que o processo para criar o CSM seria muito mais difícil, senão impossível, se não se recorresse a instrumentos como estes. Estes sistemas permitem incrementar a eficiência através da automatização de procedimentos, possibilitando a diminuição de custos operacionais[26], através de um melhor emprego dos meios.

Neste sentido, a Marinha tem vindo a desenvolver esforços nesta área, com o intuito de dotar os MRCC's com os melhores sistemas. Para isso foram estabelecidos diversos protocolos com entidades nacionais e internacionais[27]. É esta articulação que

[26] Custos operacionais incluem todos os encargos que uma empresa/instituição têm de suportar para assegurar o exercício da sua atividade.
[27] A iniciativa 5+5, originada pela França, surgiu em 1983, mas só tomou forma em julho de 1990. O objetivo desta iniciativa é promover a cooperação entre os países do Mediterrâneo ocidental, onde se incluem a França, Itália, Portugal e Espanha, aos quais se juntaram mais tarde Malta, Argélia, Líbia, Mauritânia, Marrocos e Tunísia.

permitirá à Marinha contribuir para que Portugal exerça a sua soberania sobre o território nacional.

2.1.1 *Maritime Command and Control Information System* (MCCIS)

O *Maritime Command and Control Information System* (MCCIS) é um sistema de informação que foi desenvolvido para os países membros da NATO e permite gerir informação relativa a contactos no panorama marítimo (planos sub-superfície, superfície e aéreo). A primeira versão deste sistema foi o *Alpha* CCIS (*Command and Control Information System*), que resultou de uma cooperação entre o *Supreme Allied Commander Atlantic* (SACLANT) e *U.S. Space and Naval Warfare Systems Command* (SPAWAR). O lançamento desta primeira versão ocorreu em 1992 e procurava integrar o *Navy's Joint Maritime Command Information System* (JMCIS) num ambiente C3 (*Command, Control and Communications*). Em 1996, devido às contínuas melhorias que o sistema sofreu até então e à sua proliferação, fizeram com que este se tornasse no sistema C3 da NATO, passando-se a designar, a partir daí, de MCCIS. Em janeiro de 2009 foi lançada a versão 6.0, encontrando-se atualmente na versão 6.1.0.

Este sistema foi desenhado para providenciar um sistema automático de gestão de informação capaz de auxiliar os membros da NATO no planeamento e execução de atividades militares de modo a alcançar os objetivos da aliança. O MCCIS revela-se um excelente instrumento, pois utiliza uma rede NATO segura, robusta e redundante. É através desta rede, a NATO SECRET WAN, que o sistema faz a troca de informação. À medida que a rede se expande, a nível nacional e NATO, os comandos vão-se interligando permitindo o acesso a uma maior quantidade de informação. Esta rede possibilita assim a utilização deste sistema por unidades navais, terrestre e aéreas.

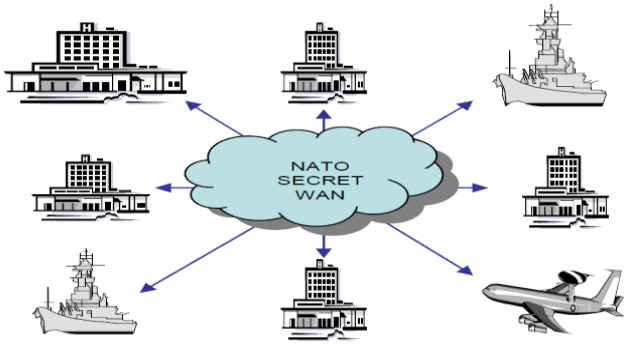


Figura 7 - NATO SECRET WAN[28].

[28] NATO (2009). *MCCIS 6.0.0 system user manual* [imagem]. [s.l.], NATO, pp1-6.

O MCCIS é um produto em constante evolução, para isso conta com o *feedback* do utilizador, assim como do administrador de área. Este sistema possibilita ao utente reportar erros encontrados e fazer pedidos de novas funcionalidades no sistema através do *Helpdesk*. Este sistema permite a aquisição automática de uma grande quantidade de informação para análise e avaliação, de forma a apoiar as decisões. O MCCIS disponibiliza ainda ao utilizador toda informação necessária para a análise do panorama. Para isso, utiliza diversas fontes de informação, de onde se destacam as mensagens OTH GOLD[29], que alimentam os diferentes módulos do sistema:

- Geográfico;
- Apoio à decisão;
- Ferramentas de planeamento;
- Comunicações táticas;
- Mensagens formatadas NATO;
- Base de dados NATO;
- Apoio a briefing;
- Funcionalidades *web*;

O módulo geográfico é usado para disseminar a informação existente sobre todos os contactos marítimos, terrestres e aéreos. Para isso este subsistema possui quatro mapas-mundo, que possibilitam a visualização de quatro áreas diferentes. Este módulo possui ainda ferramentas que permitem analisar o trânsito de um contacto. O utilizador tem também a possibilidade de decidir se deseja adicionar ou suprimir do mapa algumas informações como bases militares, portos, cidades ou fronteiras.

[29] Mensagens OTH GOLD são mensagens que têm "como finalidade reportar contatos (próprios, amigos/aliados ou não-amigos/inimigos) no âmbito da edificação do panorama marítimo o mais claro possível". (informação retirada de MARINHA (2012). *IONAV1010 – Relatos e comunicados operacionais*, Lisboa, Marinha, pp5.24.

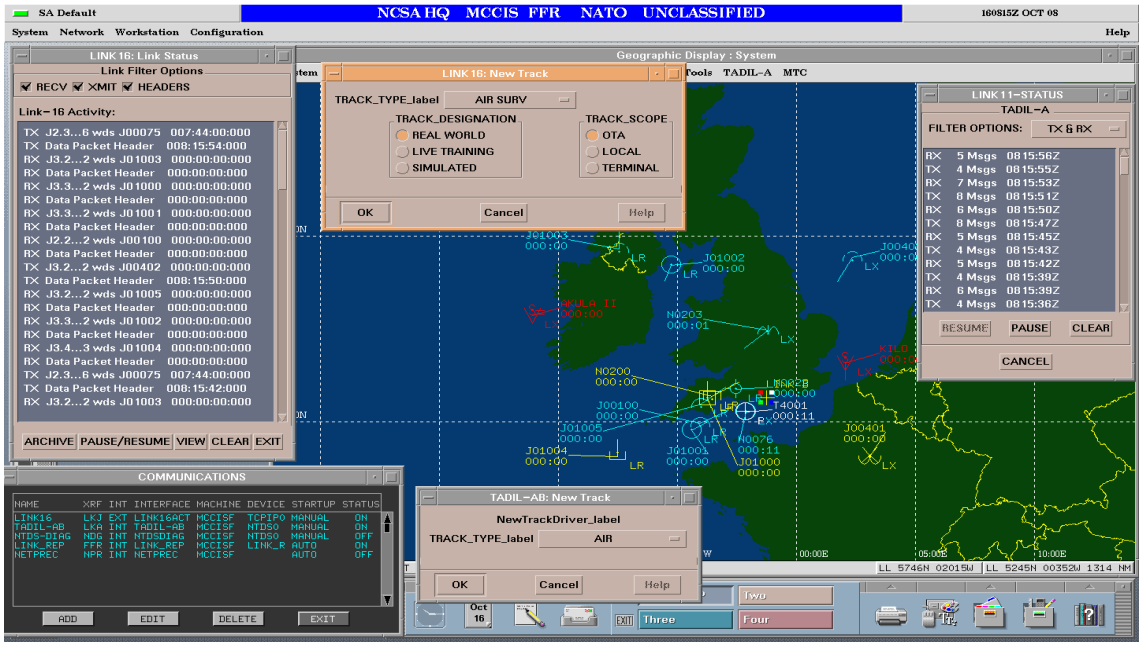


Figura 8 - Módulo geográfico.

Os módulos de apoio à decisão e ferramentas de planeamento permitem carregar as características geográficas da região, como portos, cidades e aeroportos. Estes subsistemas possibilitam também o desenho das áreas de responsabilidades e áreas de exercícios. Outro utensílio que estes módulos disponibilizam é o PIMTRACK (*Position and Intended Movement Track*), esta ferramenta permite aos utilizadores planearem o trânsito de unidades navais, terrestres e aéreas. Estes subsistemas fornecem ainda um instrumento gerador de cenários que apoia o planeamento e condução de exercícios, através da criação de unidades simuladas. Para além das ferramentas anteriormente referidas, estes módulos possuem ainda uma ferramenta de *Water Space Management*, possibilitando assim o planeamento e execução de exercícios ou operações com submarinos, contribuindo deste modo para a navegação em segurança dos mesmos.

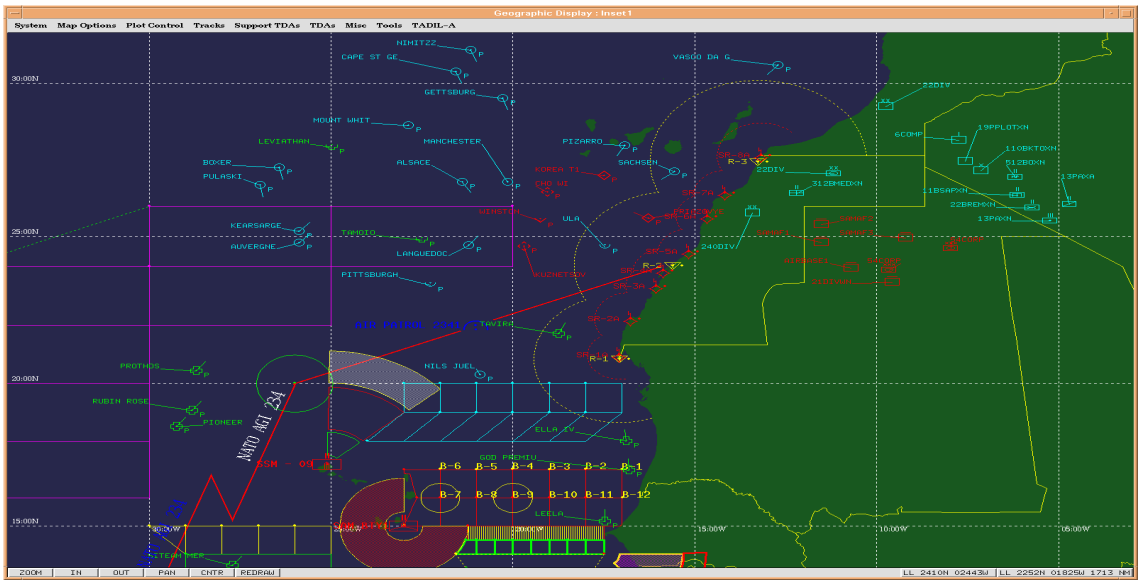


Figura 9 - Panorama após a utilização das ferramentas do módulo de ferramentas de planeamento.

O módulo de mensagens formatadas NATO, disponibiliza aos seus utilizadores a troca e processamento de informação em vários formatos e protocolos. Este módulo gera, distribui e processa mensagens formatadas, atualizando automaticamente a base de dados, através de mensagens como NAVSITREP (*Navy Situacional Report*) ou MARINTSUM (*Maritime Intelligence Summary*).

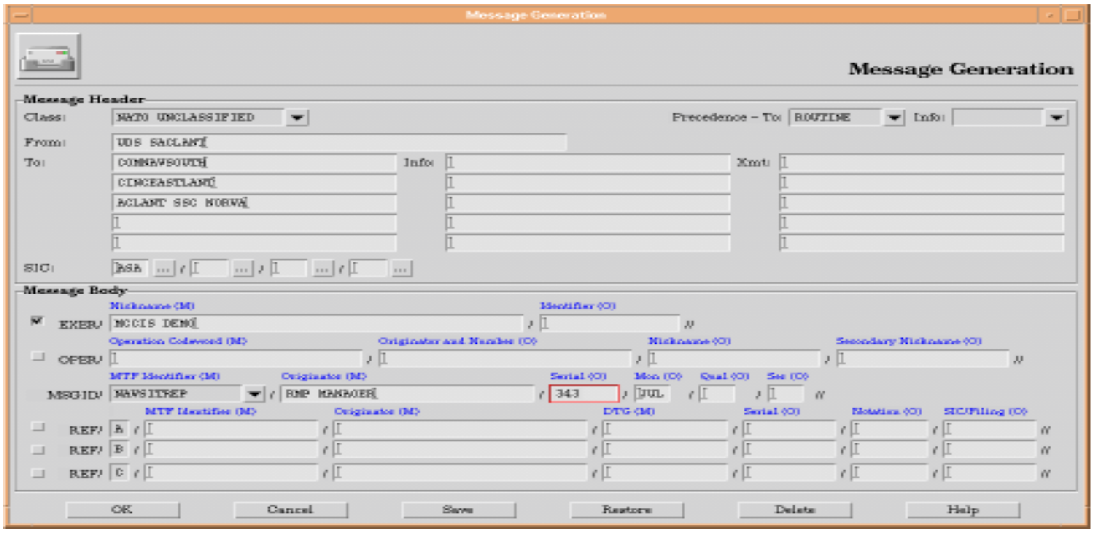


Figura 10 - Redação de uma mensagem formatada[30].

O módulo de bases de dados incorpora diversas bases de dados dinâmicas e estáticas, que podem ser atualizadas automaticamente, através dos dados provenientes das mensagens. O MCCIS inclui *links* para diversas bases de dados estáticas, de onde se destacam:

[30] NATO (2009). *MCCIS 6.0.0 system user manual* [imagem]. [s.l.], NATO, pp1-17.

- *Jane's Fighting Ship e Jane's All The World's Aircraft*[31];
- World Port Index;
- Internacional Radio Call Signs;
- MarIS (Maritime Information System).

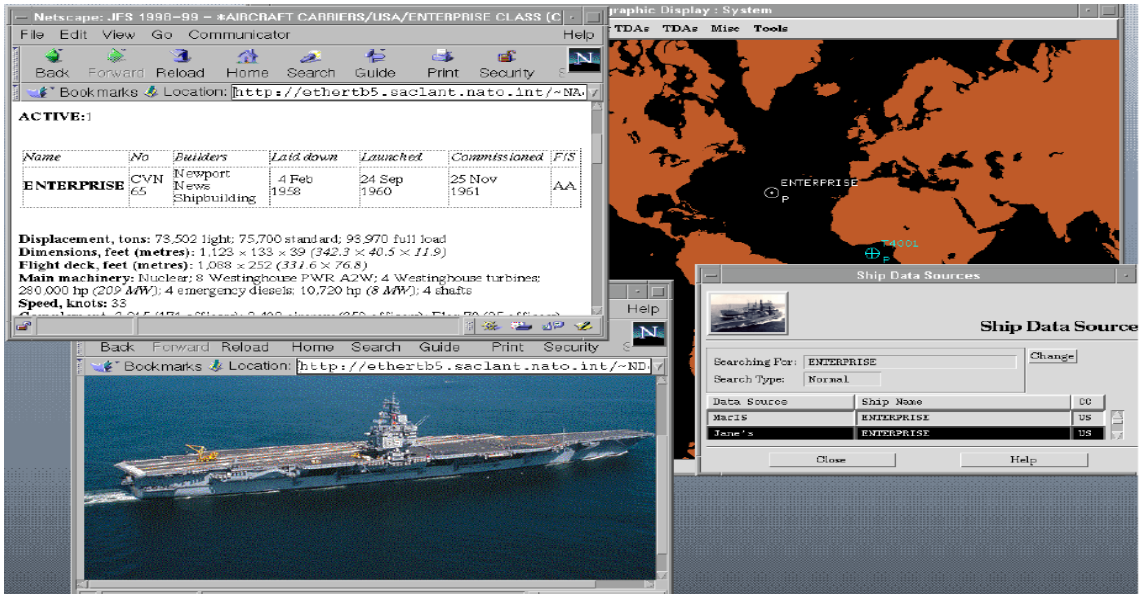


Figura 11 - Principais características do USS Enterprise[32].

O módulo de funcionalidades web íntegra tecnologias de navegação na *web*, em conjunto com um aplicativo conhecido como WISE (*Web Information Services Environment*), ampliando ainda mais o acesso ao MCCIS e reduzindo a dependência de estações de trabalho. Com o WISE, os utilizadores autorizados poderão aceder a informações como "*snapshot*" da RMP/RLP/RAP e diversos dados estatísticos residentes nos servidores MCCIS. Os utilizadores podem ainda visualizar ou carregar documentos para os servidores do MCCIS, criando assim uma rede de partilha de informação.

[31] *Jane´s Fighting Ships* e *Jane's All The World's Aircraft* são bases de dados que contêm todas as aeronaves e navios militares de todo o mundo. Estas bases de dados contêm dados sobre as características físicas dos navios (calado, boca, comprimento, deslocamento, etc) e aeronaves, assim como, o armamento e sensores.

[32] NATO (2009). *MCCIS 6.0.0 system user manual* [imagem]. [s.l.], NATO, pp1-9.

2.1.2 *TransView* (TV32)

O sistema *TransView* (TV32) é um Sistema de Informação Geográfica (SIG) que começou a ser desenvolvido em 1996, pelo centro Volpe[33]. Esta ferramenta veio dar resposta às necessidades de várias organizações envolvidas na gestão do tráfego marítimo (VOLPE CENTER, [s.d.]).

A função básica do TV32 é mostrar os contatos, incluindo AIS e radar, numa interface gráfica que permita ao utilizador ter uma visão integrada do panorama de superfície. Nesta interface os contatos podem ser sobrepostos em imagens aéreas, cartas de navegação raster[34] ou vetoriais[35]. Para além do que já foi referido, o TV32 possui ainda outras funcionalidades como:

- Cálculo de distâncias entre dois pontos;
- Cálculo do CPA, TCPA e ETA;
- Exportação de dados para o formato OTH GOLD, XML[36] ou KML[37] para serem interpretados, posteriormente, por outro programa;
- Exportação de contatos e as suas informações (rumo, velocidade, dimensões, etc.), para o *Google Earth*, através de ficheiros KML;
- Gravar e reproduzir dados guardados;
- Monitorizar fluxos de dados para a posterior elaboração de estatísticas;
- Alertar se um potencial perigo ocorrer a partir de um navio a entrar ou sair de uma zona específica definida pelo utilizador.

[33] O centro Volpe é uma agência federal pertencente ao Departamento de Transportes Norte-americano que têm como missão melhorar o sistema de transporte norte-americano, servindo como centro de excelência para a tomada de decisão informada, antecipando problemas de transporte emergentes, e avançando inovações técnicas, operacionais e institucionais.

[34] Uma carta raster é essencialmente uma imagem eletrónica da carta de papel, obtido através de um processo de digitalização rigoroso. Estas cartas têm, portanto, exatamente as mesmas informações que as cartas de papel.

[35] Uma carta vetorial possui a informação organizada por camadas, permitindo a seleção, analise e apresentação dos elementos, nela contidos, de forma personalizada ou automática.

[36] A *eXtensible Markup Language* (XML), foi desenvolvida pelo *World Wide Web Consortium* (W3C), em meados da década de 90. Esta linguagem é um dos subtipos da SGML (*Standard Generalized Markup Language*) capaz de descrever diversos tipos de dados. Tem como principal vantagem a facilidade partilha de informações através da internet.

[37] A linguagem KML (Keyhole Markup Language), foi criada pela Keyhole, Inc., que viria a ser adquirida pela Google em 2004. Esta linguagem baseia-se em XML, servindo para expressar anotações geográficas e visualização de conteúdos existentes nessa linguagem.

Figura 12 - Exportação de contatos do TV32 para o *Google Earth*[38].

Apesar de este sistema ter começado a ser desenvolvido em 1996 ainda continua, atualmente, a ser alvo de atualizações, de modo a responder às solicitações específicas das organizações que o utilizam. Como resultado das constantes atualizações, o TV32, tornou-se num sistema de confiança.

Um dos primeiros locais onde este sistema foi implementado foi o canal do Panamá, onde é usado como sistema monitorização da navegação. Atualmente o TV32 é utilizado por inúmeras organizações governamentais e privadas, nas quais se destacam a *US Coast Guard*, onde o TV32 é usado pelo Centro de Investigação e Desenvolvimento como ferramenta de desenvolvimento de protótipos. O TV32 é também usado em vários portos da América Central para monitorização da navegação, pelos pilotos do rio Colômbia como protótipo de pilotagem e monitorização da costa.

A principal vantagem deste *software* é a capacidade, que este tem, de se adaptar e personalizar, fazendo dele uma excelente ferramenta para o desenvolvimento de protótipos, que mais se adequem às necessidades do utilizador. O objetivo do TV32 é melhorar a segurança da navegação, eficiência nos canais de navegação, perceção e esclarecimento do panorama de superfície, proteção da força e a análise de dados. Neste contexto, o TV32 pode ser configurado ou reprogramado para suportar diversos requisitos operacionais. Desde da sua criação que este sistema tem vindo a ser

[38] VOLPE CENTRE (s.d.). *TV32 exported to Google Earth* [imagem]. Retirado de https://www.google.pt/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CC8QFjAA&url=http%3A%2F%2Fhandle.dtic.mil%2F100.2%2FADA519835&ei=k0llUvSVE5Kihgflj4CYAg&usg=AFQjCNGEEvkwpduMJB7de8Fn2VW6NViR-Q&sig2=YWYdx-v6C8gvfsq8rgRJ1A&bvm=bv.54934254,d.ZG4, consultado em 10 de outubro de 2013.

implementado em centros de operações marítimas, bem como numa variedade de plataformas móveis. Na Marinha, a instalação e gestão dos utilizadores desta ferramenta está a cargo da Direção das Tecnologias de Informação e Comunicação (DITIC).

2.1.3 SafeSeaNet

Atualmente, a grande maioria das trocas comerciais realizam-se por via marítima, o que faz com que todos os anos milhões de navios passem pelas águas de países membros da União Europeia. Consequentemente, centenas desses navios sofrem acidentes nas águas da União Europeia. Deste modo a informação fornecida pelos navios é fundamental para promover a segurança dos mesmos. No passado, a informação fornecida era tratada de diferentes formas, por diferentes entidades, o que fazia com que a troca de informação fosse relativamente difícil, pois eram usadas diferentes formas de armazenar, compilar e transferir a informação, o que causava incompatibilidades entre os diferentes sistemas.

Na sequência do afundamento do petroleiro Erika, na costa francesa em 1999, a União Europeia adotou várias diretivas com o objetivo de prevenir futuros acidentes e combater a poluição marítima. Estas diretivas permitiram assim ultrapassar o problema da troca de informação. Em 2003, é criada a *European Maritime Safety Agency* (EMSA) que de forma a cumprir o estipulado pela Diretiva 2002/59/EC[39], desenvolveu, em 2004, o SafeSeaNet (SSN) que só viria a ficar totalmente operacional em 2009 (EMSA, 2009).

Figura 13 - Página inicial do SafeSeaNet[40].

[39] Diretiva europeia que estabelece um sistema de monitorização da informação dos navios da comunidade europeia. Atualmente esta diretiva foi substituída pelas diretivas 2009/17 e 2011/15, estas duas últimas diretivas vieram facilitar a implementação e operação do SafeSeaNet e estenderam o campo de aplicação da diretiva até aos navios pesqueiros com mais de 15 metros de comprimento.

[40] EMSA (2013). *Safeseanet login* [imagem]. Retirado de https://portal.emsa.europa.eu/web/ssn consultado a 9 de novembro de 2013.

Em 2010, este sistema teve uma grande evolução com a incorporação de uma interface gráfica que permite a visualização de informação em cartas náuticas eletrónicas. Este *upgrade* permite ao utilizador ter, mais rapidamente, uma melhor perceção do panorama marítimo de superfície, permitindo assim tomar decisões mais rapidamente.

Esta ferramenta tornou-se no sistema europeu eletrónico de monitorização do tráfego marítimo, tendo como objetivo a melhoria da segurança e eficiência do tráfego marítimo, aperfeiçoando a resposta das autoridades a acidentes ou a situações potencialmente perigosas (incluindo operações SAR), contribuindo também para a melhoria da prevenção e deteção de navios poluentes.

O SSN tem o intuito de tornar o mar mais seguro. Para isso, disponibiliza aos seus utilizadores, um conjunto de ferramentas que possibilita a identificação antecipada dos navios de elevado risco, melhorando a resposta em casos de emergência ou de poluição. Este sistema tem também como objetivo o aumento da eficiência através da uniformização de acesso a dados, fornecimento de ajuda aos utilizadores e aumento da eficiência da logística nos portos. Outro dos objetivos é o aumento da qualidade de monitorização da União Europeia, através do fornecimento de informação mais precisa da localização dos navios.

A ferramenta SSN possui uma função chamada Relatório de Incidentes. Esta função mostra ao utilizador os navios ou cargas que representam um potencial risco para os Estados Membros, assim como informações sobre os perigos que carga representa, a sua rota e informação sobre acidentes passados.

Os dados recolhidos são armazenados num único servidor, o que faz com que seja simplificado o modo dos utilizadores do SSN obterem a informação. O acesso concedido aos utentes do SSN depende da autoridade nacional com competências nessa área, que gere os direitos de acesso a nível nacional.

O SSN recebe os seguintes tipos de mensagens: relatório de incidentes, notificação de porto, navio e materiais perigosos. O relatório de incidentes é usado para notificar o SSN que o fornecedor de dados detém informação sobre um incidente específico. A notificação de um navio é usada para fornecer ao SSN os detalhes da viagem desse navio e a informação da carga. Estas notificações têm como base os dados fornecidos por AIS e MRS[41] (*Mandatory ship Reporting System*). A notificação de porto é uma

[41] Mensagens enviadas, pelos comandantes dos navios, para estações em terra, contendo informação sobre a identificação do navio, rumo, velocidade e carga.

mensagem enviada por um porto a informar o SSN que um determinado navio encontra-se a poucas milhas desse porto. Nessa mensagem é incluída a hora a que está prevista a chegada (ETA) e o número de pessoas a bordo, assim como o tipo de carga que transporta. A notificação de materiais perigosos é usada para informar o SSN que um determinado navio transporta materiais perigosos. Nesta notificação é também disponibilizada informação detalhada sobre os respetivos materiais[42].

O SSN foi estabelecido como uma plataforma europeia para a troca de informação, ligando as autoridades e representando um avanço significativo em termos de segurança marítima. Este instrumento disponibiliza informação desde caracter mais geral (área da União Europeia) até ao mais particular (cais de um porto). O sistema SSN permite ainda a visualização do histórico de posições e informações de um navio ou de uma determinada carga em particular. A informação é apresentada em cartas náuticas eletrónicas, contendo uma grande variedade de informações como exemplo esquemas de separação de tráfego, faróis, entre outros.

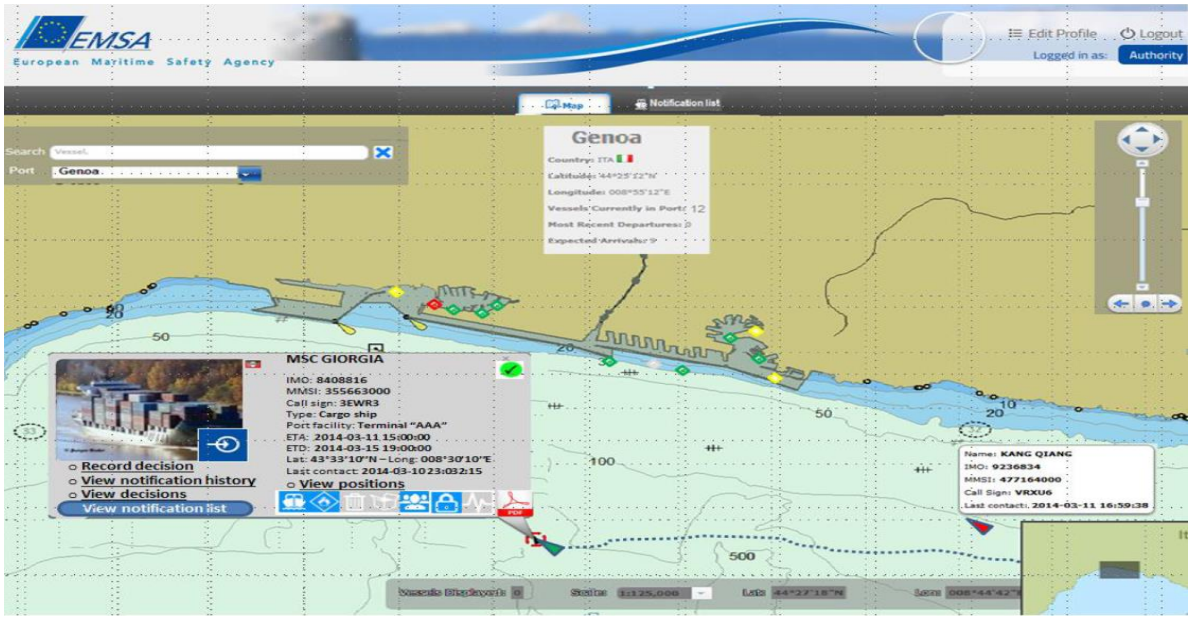


Figura 14 - Interface gráfica do SafeSeaNet.

2.1.4 Sistema de Apoio à Decisão para a Atividade de Patrulha (SADAP)

O Sistema de Apoio à Decisão para a Atividade de Patrulha (SADAP) é uma ferramenta, desenvolvida pela DAGI em 2006, que efetua uma análise da atividade de fiscalização marítima realizada pelos navios da Marinha e unidades da Direção Geral da Autoridade Marítima (DGAM). Este sistema tem como principal objetivo criar um

[42] EMSA (2009). *An information system to improve maritime safety in Europe*. [s.l.], EMSA.

processo automático de troca de informação entre o Comando Naval (COMNAV) e a DGAM, no que diz respeito aos resultados obtidos nas fiscalizações marítimas.

Esta ferramenta, como já referido, veio tornar possível a troca de informação automática entre o COMNAV e DGAM, eliminando uma das lacunas existentes na gestão da informação da atividade da fiscalização marítima. Este sistema destina-se essencialmente a auxiliar as atividades de fiscalização marítimas, principalmente na fase de planeamento de ações fiscalização, contribuindo, deste modo, para o aumento da eficácia e eficiência no emprego dos meios.

Inicialmente o SADAP apenas disponibilizava estatísticas sobre as atividades de fiscalização da pesca. Mas desde logo tornou-se evidente que o sistema poderia ser explorado noutras vertentes. Até ao ano de 2008 foram acrescentados ao sistema SADAP novos módulos, fazendo com que atualmente seja constituído por seis módulos: módulo de análise da atividade de pesca, módulo de análise de fiscalização, módulo de busca e salvamento marítimo, módulo de capacidade AIS, módulo de elaboração de mensagens formatadas, e módulo de regras, malhagens e espécies.

O módulo de análise da atividade de pesca permite ao utilizador inserir no sistema dados relativos ao posicionamento e movimento das embarcações de pesca que detenham equipamentos de monitorização continua (equipamento MONICAP), deste modo é assim possível analisar as trajetórias utilizadas pelas embarcações de pesca, assim como determinar as áreas onde existe uma maior densidade de embarcações.

No que diz respeito ao módulo de análise de fiscalização, este confere ao utilizador a possibilidade de consultar dados provenientes de mensagens relativas a fiscalizações anteriormente realizadas pelas capitanias e unidades navais. Este módulo permite ainda criar um planeamento de missão, apoiado numa base de dados referente a fiscalizações anteriores, contribuindo deste modo para o aumento da eficácia e eficiência.

O módulo de busca e salvamento marítimo permite ao utilizador calcular áreas de busca e desenhar os planos de busca utilizando o método retangular e quadrado expansivo[43]. Este subsistema do SADAP inclui ainda, desde 2011, uma aplicação que permite ao utilizador a visualização, criação e edição de alertas SAR.

O módulo de capacidade AIS surgiu da necessidade de criar um subsistema do SADAP que permitisse ter o panorama de superfície da costa portuguesa esclarecido e

[43] Os métodos da deriva, fiadas retangulares e quadrado expansivo (normalmente só utilizado quando uma aeronave se encontra destacada para as operações de busca e salvamento) têm por base a doutrina NATO, descrita na publicação ATP 10 (D).

em constantemente atualizado. Para isso utilizou-se os dados fornecidos pelo sistema AIS, pois é cada vez maior o número de embarcações que dispõem deste sistema. Este módulo permite ainda ao utilizador aceder a uma base de dados de contatos de interesse e visualizar as rotas mais utilizadas pelos navios, pois tem como base o histórico dos trajetos desses mesmos navios.

O módulo de elaboração de mensagens formatadas auxilia o utilizador na redação de mensagens como ATA, ATD[44], autos de notícia, FISCREP[45], NAVSIT[46], permitindo assim redação da mensagem de forma mais rápida.

Por último o módulo de regras, malhagens e espécies, que auxilia o utilizador durante uma ação de fiscalização marítima. Este subsistema do SADAP permite o cruzamento de informação, com base na legislação em vigor, no que respeita à malhagem das embarcações, espécies que se encontram proibidas de pescar, espécies-alvo, áreas e períodos de defeso. Desta forma é possível identificar automaticamente presumíveis infrações.

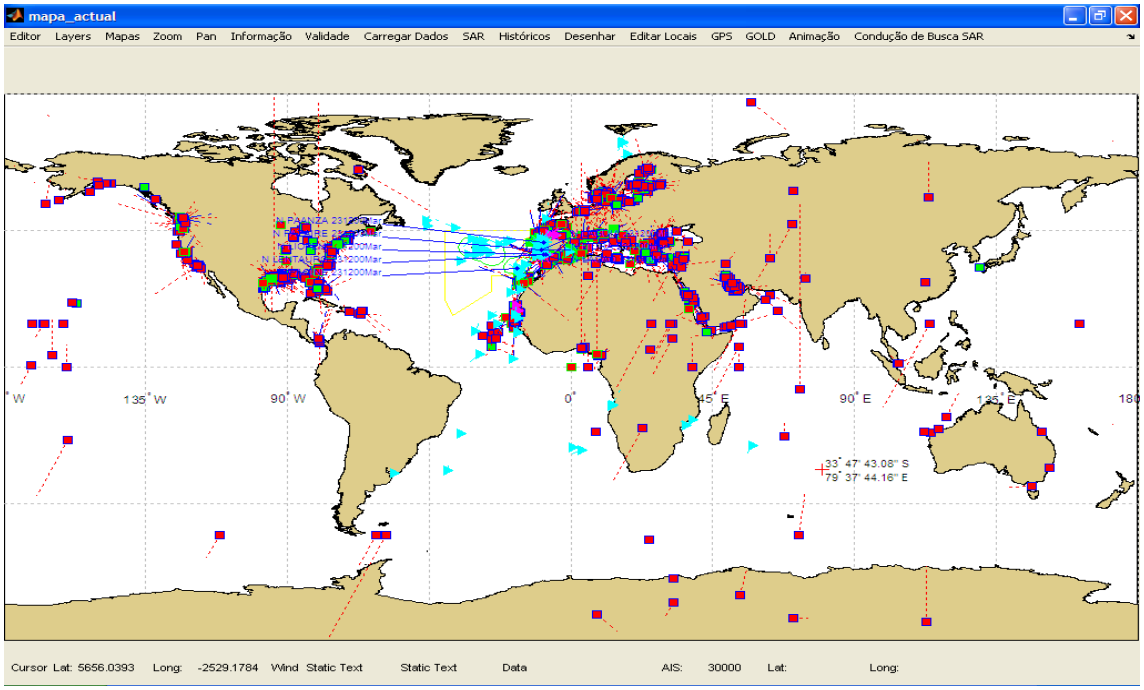


Figura 15 – Panorama AIS e MONICAP na ferramenta SADAP.

Atualmente, o sistema SADAP tem vindo a revelar-se cada vez mais útil para a DGAM e COMNAV, permitindo a troca automática de informação entre estas duas entidades. Desta forma é possível atuar mais rapidamente sendo mais eficaz e eficiente.

[44] Comunicados de movimentos a efetuar imediatamente após a largada e chegada respetivamente, relatando a hora exata a qua a mesma ocorreu.
[45] Comunicado que se destina a relatar o resultado da inspeção efetuada a uma embarcação de pesca ou de recreio, nacional ou estrangeira, em atividade nas águas sob soberania ou jurisdição nacional.
[46] Comunicado de situação a enviar periodicamente pelos Comandos de Forças.

Para além do que já foi referido, os subsistemas do SADAP fornecem apoio ao utilizador em diversas áreas, que vão desde o simples auxílio na redação de uma mensagem até ao cálculo de áreas de busca em operações SAR. Em suma, o SADAP permite à DGAM e COMNAV prestarem um melhor serviço público, efetuando, assim, uma melhor gestão dos seus recursos.

2.1.5 *Marine Traffic*

O *Marine Traffic* é um sistema de CSM *online* e gratuito, em tempo real, que surgiu em 2007, no âmbito de um projeto académico liderado pelo professor Dimitrios Lekkas da Universidade Egeu na Grécia[47]. Este *site* é parte de um projeto baseado numa comunidade aberta. Este sistema dedica-se à recolha e apresentação de dados, que são explorados em diversas áreas de investigação, tais como:

- Estudo das telecomunicações marítimas e parâmetros de propagação, tendo vista o aumento da eficiência;
- Simulação de movimentos dos navios, tendo em vista a melhoria da segurança da navegação e atuação em caso de incidente;
- Sistemas de informação interativo;
- Bases de dados que fornecem informações em tempo real;
- Processamento estatístico do tráfego marítimo nos portos;
- Conceção de modelos para a determinação da origem da poluição;
- Projeto de algoritmos eficientes para avaliação do trajeto marítimo e determinação do ETA;
- Correlação da informação recolhida com os dados meteorológicos;
- Cooperação com institutos dedicados à proteção do meio ambiente.

Esta aplicação fornece informação ao público, em tempo real, de forma gratuita, sobre os movimentos de navios e portos de todo o mundo. A recolha de dados é baseada numa rede de estações que estão equipadas com um recetor AIS, um computador e uma ligação à internet. A unidade AIS recebe dados, que são processados por um programa específico instalado no computador e, em seguida, enviados para uma base de dados, através da internet. O *Marine Traffic* por se tratar de um serviço gratuito, incentiva os seus utilizadores a construírem a sua própria estação, contribuindo assim para o

[47] UNIVERSIDADE EGEU (s.d.). *Dimitrios Lekkas*. Retirado de http://www.syros.aegean.gr/users/lekkas/, consultado a 12 de agosto de 2014.

aumento da rede de estações e área de cobertura. Os recetores AIS transmitem nos seus dados três tipos de informações básicas:

1) Informações dinâmicas, como a posição do navio, velocidade, estado de navegação atual (a navegar, fundeado, atracado), e rumo.
2) Informações estáticas, como o nome do navio, número IMO, MMSI e dimensões.
3) Informações específicas da viagem, como o destino, ETA, tipo de carga transportada e calado do navio.

Os dados recebidos são carregados na base de dados em tempo real, ficando imediatamente disponíveis no mapa. No entanto, algumas das posições indicadas no mapa podem não ser continuamente atualizadas, pois os navios podem encontrar-se fora do alcance. O sistema *MarineTraffic* não cobre todos os oceanos e mares do mundo, como podemos verificar na figura seguinte, mas apenas áreas costeiras específicas, onde um recetor AIS em terra está instalado.

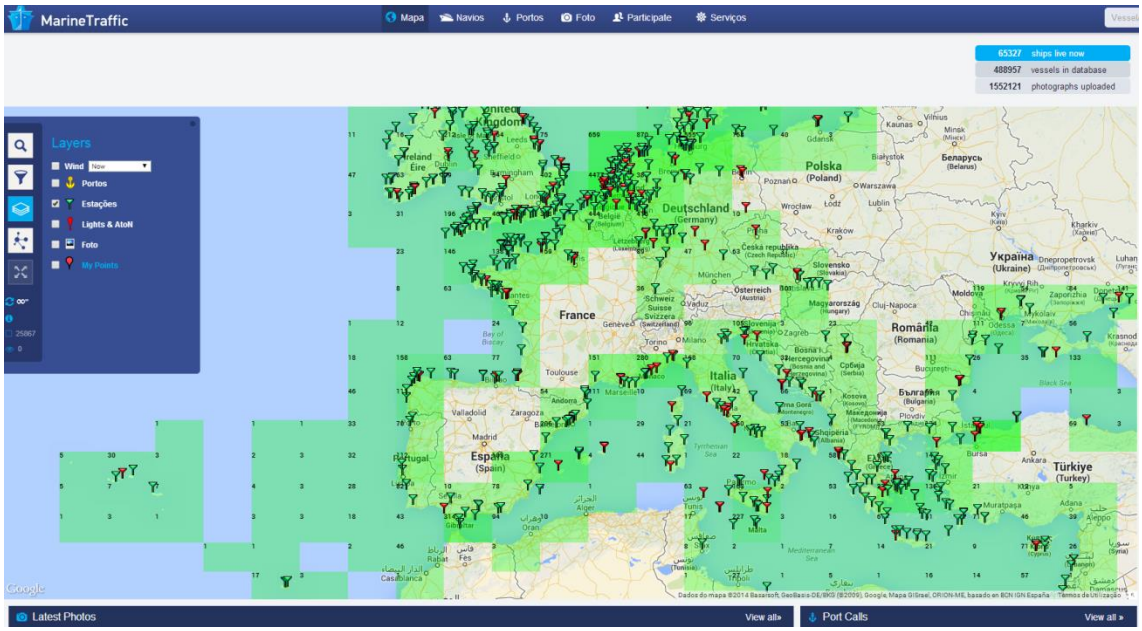


Figura 16 - Área coberta pelo *Marine Traffic* na Europa[48].

O *Marine Traffic* é uma ferramenta exclusivamente baseada na informação AIS enviada pelos navios. Consequentemente, a configuração correta dos transmissores AIS é imprescindível para o bom funcionamento do sistema. Este facto não se verifica em diversos casos, pelo que, o sistema apenas deve ser usado para esclarecimento do

[48] MARINE TRAFFIC (2014). *Marine Traffic*. Retirado de https://www.marinetraffic.com/pt/ais/home/?lang=pt, consultado a 2 de maio de 2014.

panorama em caso de dúvida, e nunca com aspetos relacionados com a segurança da navegação. A informação fornecida pode estar incompleta, obsoleta ou conter erros e não podem substituir os equipamentos de segurança a bordo. Embora o *Marine Traffic* não se destine a ser usado como uma ferramenta de segurança, há muitos casos em que se os navios dessem a sua posição a conhecer poderia melhorar a segurança da navegação.

Atualmente é composta por apenas dois módulos: o módulo geográfico e o módulo base de dados. O módulo geográfico utiliza o mapa Google para mostrar os navios ao utilizador. Os navios constantes no mapa estão equipados com um transmissor AIS e navegam dentro do alcance de receção de um recetor AIS instalado em terra. Este subsistema possibilita ao utilizador a aplicação de diversos filtros como: nome do navio, portos, faróis, estações AIS, vento, entre outros.

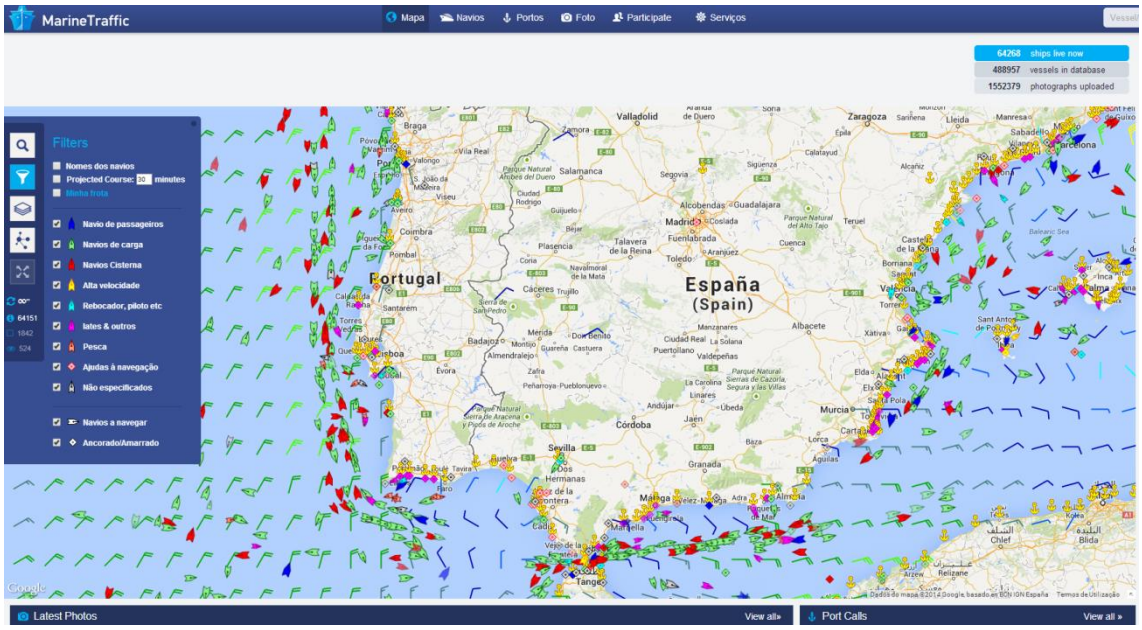


Figura 17 - Panorama marítimo na costa da península Ibérica às 19:30 de dia 2 de maio de 2014[49].

Para uma melhor e mais rápida perceção do panorama os navios foram representados com cores diferentes e separados em várias categorias como: navio de passageiros, carga, cisterna, alta velocidade, pesca, rebocador, piloto, iate e não especificados. Este módulo disponibiliza ainda ao utente mapas de densidade, onde pode ver a densidade de tráfego de forma geral ou uma categoria em específico.

[49] MARINE TRAFFIC (2014). *Marine Traffic*. Retirado de https://www.marinetraffic.com/pt/ais/home/?lang=pt, consultado a 2 de maio 2014.

Figura 18 - Densidade do tráfego marítimo na Europa no dia 2 de maio de 2014[49].

O módulo base de dados permite ao utilizador fazer uma pesquisar por navio ou por porto. Esta base de dados contém cerca de 489.000 navios e 14.000 portos. Este subsistema disponibiliza ao utente diversas formas de pesquisar navios, tais como: nome, bandeira, tipo de navio, comprimento, entre outras. Ao selecionar um navio este módulo mostra ao utilizador a informação estática, dinâmica e de viagem proveniente do AIS.

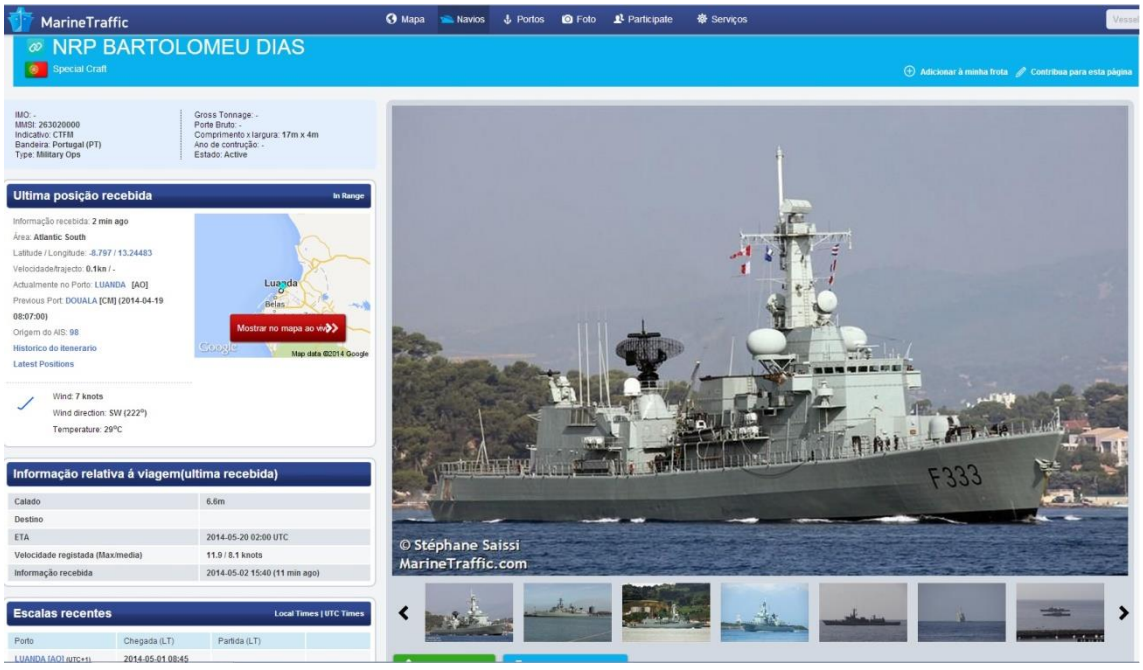


Figura 19 - NRP Bartolomeu Dias na base de dados do *Marine Traffic*[50].

[50] MARINE TRAFFIC (2014). *NRP Bartolomeu Dias*[imagem]. Retirado de https://www.marinetraffic.com/pt/ais/details/ships/263020000/vessel:NRP_BARTOLOMEU_DIAS, consultado a 2 de maio de 2014.

Esta ferramenta apesar de possuir menos funcionalidades do que as outras até aqui mencionadas revela-se extremamente útil, pois fornece em tempo real o posicionamento dos navios. O facto de esta aplicação ser bastante simples, de fácil utilização e gratuita, fez com que, desde da sua criação, tivesse uma rápida proliferação de utilizadores em todo o mundo.

2.1.6 IMDatE

Na sequência da adoção da política de integração marítima da União Europeia, em 2009, e várias resoluções do Conselho Europeu sobre a integração da informação de notificação de navios, a EMSA iniciou o projeto de integração dos seus sistemas marítimos numa só plataforma operacional mais poderosa e flexível, designada de *Integrated Maritime Data Environment* (IMDatE). No futuro, este sistema vai combinar e processar dados de aplicações marítimas da EMSA (SafeSeaNet, CleanSeanet[51], EU LRIT DC[52], THETIS[53]) e outras fontes externas para prestação de serviços mais abrangentes e configuráveis para os utilizadores, bem como apoio a retransmissão de dados entre os próprios aplicativos. Esta aplicação tem como objetivos melhorar o CSM para os Estados Membros e apoiar a troca e partilha de informações entre as diferentes aplicações[54].

O IMDatE pretende apoiar e melhorar as aplicações da EMSA já existentes, apesar destas já serem amplamente utilizadas pela comunidade marítima. Estas melhorias permitiram ao utilizador mais opções de visualização de dados, novas interações na ligação máquina-a-máquina, e monitorização automática do comportamento do navio. A integração das diferentes aplicações numa só fará com que passe a ser necessário efetuar *login* apenas uma vez deixando de ser necessário fazer login em cada uma das aplicações. A verificação dos dados vai também melhorar a qualidade dos mesmos no

[51] CleanSeaNet é um sistema baseado em satélites da União Europeia para a deteção de derrames de petróleo no mar usando imagens SAR (*Satellite Aperture Radar*).
[52] O *European Long Range Identification and Tracking Data Centre* (EU LRIT DC) é uma aplicação que usa as comunicações satélites para seguir todos os navios de países membros da União Europeia por todo o mundo, bem como qualquer navio, independentemente da sua bandeira, desde que não se encontrem a mais de 1000 NM da costa da União Europeia.
[53] THETIS é uma aplicação, com base na internet, que disponibiliza informação relacionada com inspeções a navios, reportando-as a todos os controlos de portos de toda a Europa.
[54] EMSA (2014). *Integrated Maritime Data Environment*. Retirado de http://emsa.europa.eu/operations/maritime-monitoring/86-maritime-monitoring/1520-integrated-maritime-data-environment-imdate.html, consultado a 6 de fevereiro de 2014.

sistema IMDatE. Esta verificação é feita através da confirmação dos detalhes dos navios em diferentes registros.

O acesso à plataforma IMDatE pode ser concedido via internet, via interface de sistema para sistema com base em XML, e outras formas de exportação de dados, tais como correio eletrónico, entre outras. A plataforma é construída para armazenar sistematicamente e fundir os relatórios de posição do navio, correlacionando-os com os navios alvos detetados a partir de imagens de satélite ou radares costeiros, e informação sobre navios disponibilizada pelas aplicações e bases de dados da EMSA, de modo a fornecer uma imagem completa do panorama de superfície.

Neste contexto, o IMDatE permite filtrar os dados, o que, atualmente, representa uma grande vantagem, dada a grande quantidade de dados que chegam todos os dias aos centros. Esta aplicação possibilita também a deteção de ausência de comunicações por parte de um navio, permitindo assim despistar possíveis situações de perigo. Para além do que já foi referido, este sistema permite ainda resumir relatórios, adicionar informação operacional, detetar informação errónea, e remover inconsistências.

Além da imagem de tráfego marítimo, quase em tempo real, o IMDatE fornece ferramentas de análise para avaliar, detalhadamente, as trajetórias dos navios ou eventos específicos associados a um ou mais navios. A plataforma do IMDatE possui um mecanismo de monitorização automatizado que alerta o utilizador caso um navio saia ou entre em áreas sensíveis, encontre-se no mar com outro navio, mude repentinamente o porto de escala, ou desvie-se do planeamento.

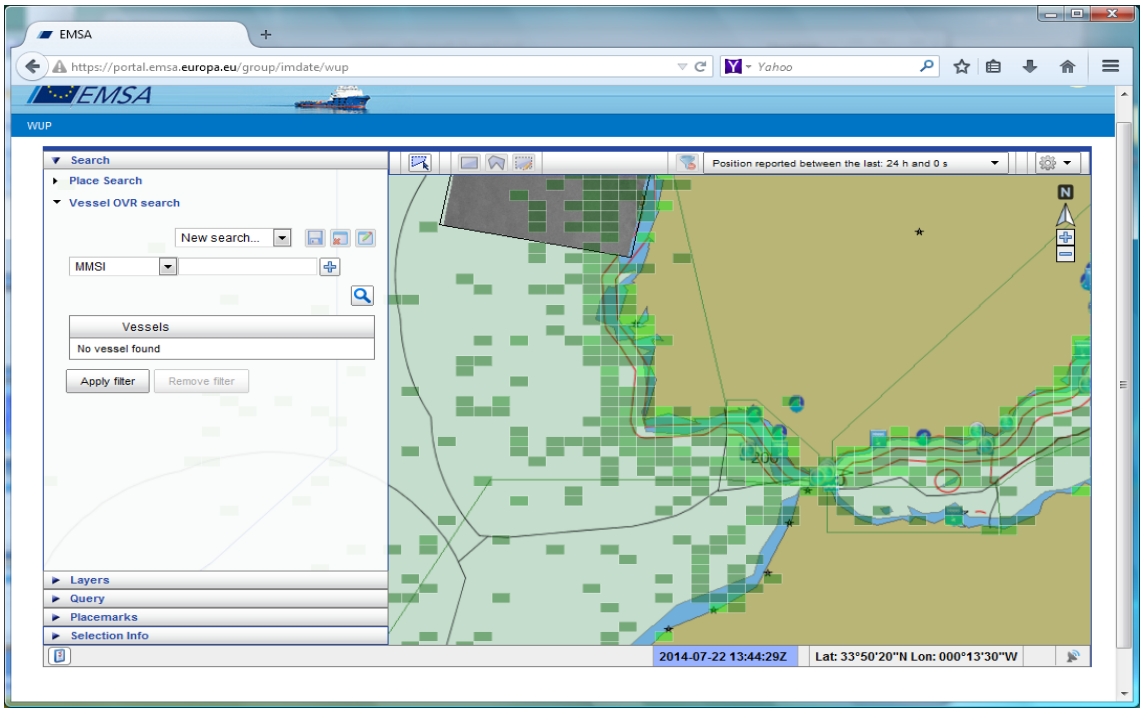


Figura 20 - Panorama de superfície da costa portuguesa no dia 22 de julho de 2014.

Vários projetos-piloto têm sido estabelecidos para explorar a possibilidade de adicionar novas fontes de dados ao quadro marítimo já existente. Nas novas fontes de dados destacam-se os sistemas de localização dos navios via satélite (VMS) e dados AIS via satélite. A plataforma IMDatE vai criar um centro de processamento de dados específico para AIS via satélite, facilitando o processamento e distribuição de outras novas fontes de dados.

A integração das diferentes aplicações da EMSA num único sistema, permite aos utilizadores monitorizarem na sua área de interesse, com um só programa, os navios que se encontram a pescar, poluir ou fora do planeamento. Este sistema proporciona assim aos seus utilizadores o esclarecimento do panorama de superfície, quase tempo real. Este fato vem contribuir para a melhoria da segurança da navegação, prevenção de acidentes e combate à poluição.

2.1.7 *Oversee*

O projeto *Blue Eye* surgiu da necessidade de superar os desafios relacionados com a eficácia e eficiência das missões de busca e salvamento marítimo (SAR), fiscalização marítima e proteção do ambiente. Este projeto tem o intuito de melhorar a capacidade de resposta da Marinha em caso de ações SAR ou catástrofes ambientais, sendo para isso necessárias soluções tecnológicas inovadoras. O consórcio do projeto constituído

pela empresa *Critical Software*, Marinha e Faculdade de Engenharia da Universidade do Porto (FEUP), iniciou os trabalhos em abril de 2011, estimando que os mesmos estejam concluídos em 2014. Este projeto é financiado pelo Fundo Europeu de Desenvolvimento Regional (FEDER), via Quadro de Referências Estratégico Nacional (QREN), e conta com um investimento total de 1.732.929,32€[55].

O sistema *Oversee* surge assim no âmbito do desenvolvimento do projeto *Blue Eye*, tendo como objetivo investigar e desenvolver tecnologia que irá aumentar a eficiência e eficácia na segurança marítima, operações de proteção ambiental e fiscalização marítima. Pretende-se assim desenvolver um sistema de informações na área do CSM que garanta o apoio às operações da Marinha, fornecendo para isso um melhor esclarecimento do panorama de superfície, possibilitando assim uma melhor antecipação de riscos, coordenação com entidades externas, que leva a uma maior capacidade de planeamento de operações e consequente otimização de recursos e custos, levando à prestação de um melhor serviço público.

No desenvolvimento deste projeto surge a oportunidade de desenvolver e validar soluções tecnologicamente inovadoras, com foco em dois eixos distintos. O primeiro vai explorar a aplicação das Tecnologias de Informação como um meio de superar o problema da integração dos diferentes sistemas, facilitando assim a troca de informação entre diferentes entidades nacionais e internacionais, melhorando o quadro operacional e processo de decisão. O segundo irá explorar novos conceitos tecnológicos, com o objetivo de avaliar a sua aplicabilidade às operações marítimas.

À semelhança de outros sistemas, o *Oversee* sobrepõe a informação adquirida a partir de diversas fontes de dados numa carta náutica eletrónica, possibilitando deste modo a visualização geral do tráfego marítimo. O *Oversee* integra dados de tráfego marítimo a partir de fontes como o AIS, SAT- AIS, VTS[56], e sistemas de posicionamento SATCOM como LRIT ou MONICAP. Esta aplicação disponibiliza ao utilizador informação meteorológica (velocidade e direção do vento, temperatura do ar, nebulosidade e precipitação), oceanográfica (direção e altura significativa[57] das ondas, período de pico, temperatura da superfície do mar, marés e correntes), cartográfica (portos, batimetria, plataforma continental e áreas de jurisdição como ZEE, águas

[55] CRITICAL SOFTWARE (2013). [s.n.]. Retirado de http://www.criticalsoftware.com/rd/, consultado a 10 de outubro de 2013.
[56] VTS – *Vessel Traffic Service* é responsável pela gestão da navegação dentro da sua área de jurisdição.
[57] A altura significativa representa a média do terço superior de todas as ondas analisadas durante o período de tempo definido, geralmente representada em metros.

territoriais e áreas SAR). Para além do que já foi referido, o *Oversee* fornece ao seu utilizador várias ferramentas de apoio à decisão como: análise de desvio, análise de rotas, mapas de risco e diagramas de impacto. Este sistema também permite a correlação e fusão de informação de fontes de dados ativos e históricos. Desde de Setembro de 2012 que o sistema *Oversee* se encontra em fase de teste e validação operacional no COMAR.

A aplicação *Oversee* é constituída por três módulos: o módulo de fiscalização marítima (*Sea Law Enforcement*), módulo de proteção ambiental (*Environmental Monitoring and Protection*) e módulo de busca e salvamento (*Search and Rescue*)[58].

No módulo de fiscalização marítima, esta ferramenta marca na carta náutica eletrónica o local onde foram efetuadas as últimas fiscalizações. Ao selecionar-se o ícone de uma fiscalização é possível consultar informações como: navio que realizou a fiscalização, navio fiscalizado e resultado da fiscalização (Figura 21). No separador *Inspection Cases* é possível consultar um dossier eletrónico com as fiscalizações efetuadas, sendo também disponibilizada informação sobre o caso.

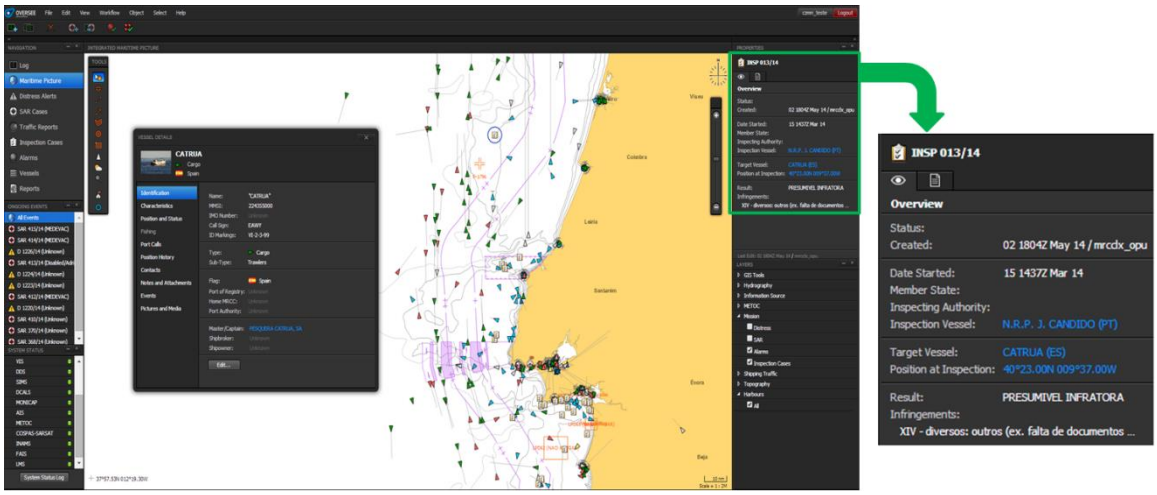


Figura 21 - Fiscalização marítima efetuada pelo NRP Jacinto Cândido à embarcação Catrua.

O módulo de proteção ambiental integrar diversas fontes de dados como boias *in situ* e veículos autónomos, entre outros. Este subsistema do *Oversee* fornece uma imagem marítima integrada, que reúne informações sobre o ambiente a partir de diversas fontes, apoiando a prevenção da poluição, permitindo a deteção da mesma, e sustentação do planeamento de resposta, execução e análise.

[58] CRITICAL SOFTWARE (2014). *Intelligent Maritime Operations Centre*. Retirado de http://www.criticalsoftware.com/uploads/resources/20140331%20-%20Oversee%20-%20Flyer%20A4_20140403155214.pdf?v34, consultado a 31 de março de 2014.

Por último o módulo de busca e salvamento tem como função primária a integração de alertas de fontes como COSPAS -SARSAT, DSC[59] ou rádio, georreferenciando-os numa carta náutica eletrónica. À semelhança do módulo de fiscalização marítima, após se selecionar um acontecimento SAR são fornecidas informações sobre o mesmo (Figura 22). No separador SAR *Cases* é fornecido, ao utilizador, um dossiê eletrónico com os últimos casos SAR.

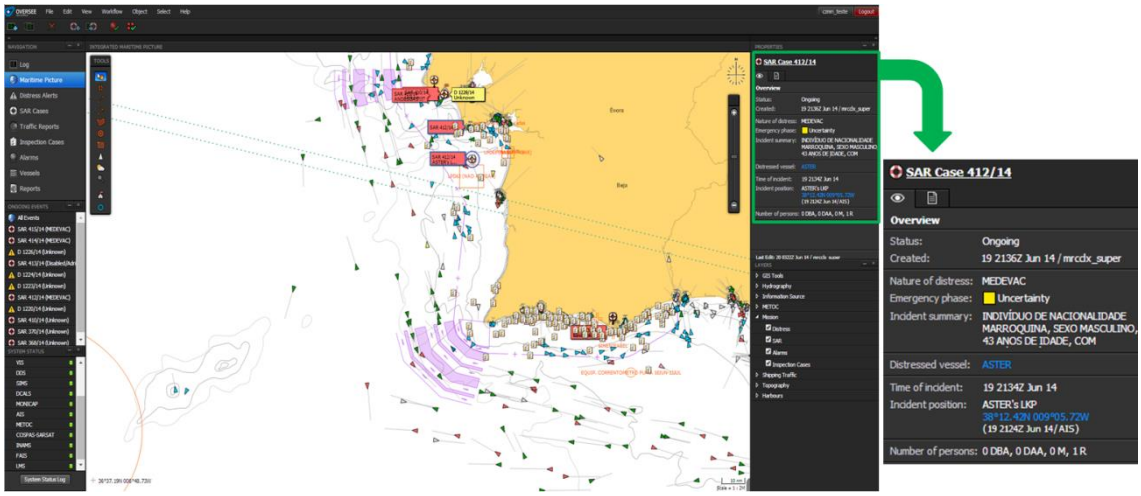


Figura 22 - Caso SAR 412/14, evacuação médica de um elemento da guarnição da embarcação Aster.

Esta aplicação veio reforçar as capacidades da Marinha, permitindo um melhor esclarecimento do panorama de superfície, elaboração de avisos antecipados, gestão de incidentes e apoio a planeamento de missões. Deste modo, revela-se uma excelente ferramenta permitindo aquisição de novas valências até então inexistentes.

2.2 AISINTEL e trabalho precedente

O protótipo AISINTEL é uma ferramenta desenvolvida em MATLAB, pela DAGI, que teve a sua origem em dissertações de mestrado de alunos da Escola Naval (Filipe J.; Melo L., 2010), orientadas pelo 1TEN Gonçalves de Deus.

Este protótipo tem vindo a ser gradualmente incrementado com funcionalidades de *Intelligence*[60] resultantes de pedidos e necessidades provenientes da comunidade operacional e também de entidades externas à Marinha ligadas às atividades marítimas. Atualmente, o AISINTEL integra nove módulos distintos, em que cada um destes

[59] Digital Selective Calling (DSC) é um sistema de envio de mensagens pré-definidas em média frequência (MF), alta frequência (HF) e muito alta frequência (VHF). Este sistema faz parte do Global Maritime Distress Safety System (GMDSS).

[60] Intelligence – O produto resultante da recolha, processamento, integração, análise, avaliação e interpretação das informações disponíveis sobre os outras nações ou áreas externas.

agrupa um conjunto distinto de funcionalidades que tiram partido dos dados AIS. O desenvolvimento destes módulos pode ser agrupado por ano:

Tabela 1 - Evolução do protótipo AISINTEL.

Ano	Módulo	Dissertações de mestrado
2010	Pesquisa em área	Melo L., 2010 e Filipe J., 2010
	Pares de navios	Melo L., 2010 -
	Análise de trajetória	Melo L., 2010 -
	Trajetórias Simultâneas	-
2011	Análise de Padrões	Melo H., 2011
	Base de dados de navios	-
2012	Planeamento de patrulha	Gomes C., 2012
	AIS SAR	-
2013	Análise de Incidentes	-
2014	Alarmística	Fernandes P., 2014

Um dos principais objetivos que esteve na origem do protótipo AISINTEL, foi a oportunidade de iniciar o registo contínuo de dados AIS e MONICAP, tendo em vista o desenvolvimento de estudos de Investigação Operacional (IO). Esta ferramenta veio também possibilitar o cumprimento de outro objetivo: o desenvolvimento de alertas utilizando dados AIS. Este último objetivo é concretizado com a presente dissertação, pois pela primeira vez, os dados AIS são utilizados para definir alertas auxiliando, deste modo, o utilizador a identificar efetuais situações de perigo ou incongruências em termos de padrão de comportamento.

Os dados AIS, provenientes de países que contribuem com o seu panorama AIS para o MSSIS, são recolhidos pelo CCMAR Northwood (antes eram coligidos pelo CCMAR Nápoles), e disponibilizados posteriormente, através da internet, sendo gravados no servidor AIS da DITIC, donde são retirados pela DAGI para o subsequente processamento. Este procedimento encontra-se a ser executado desde de janeiro de 2010, de forma automática e continua (Deus R., 2011).

O protótipo AISINTEL permite a análise e visualização de dados AIS e MONICAP através de um conjunto de módulos, sendo que estes, agrupam algoritmos para realizar a mesma função ou funções semelhantes. Atualmente esta ferramenta é constituída por

nove módulos: plano de patrulha, operações SAR, pesquisa em área, análise de padrões, pares de navios, análise de trajetória, trajetórias simultâneas, análise de incidentes e base de dados de navios.

Figura 23 - Interface inicial do AISINTEL a 11 de agosto de 2014.

2.2.1 Módulo pesquisa em área

O trabalho desenvolvido nas dissertações de mestrado mencionadas anteriormente veio dar resposta a um conjunto de requisitos operacionais identificados pela DAGI e aprovados pelo COMNAV. Os requisitos operacionais foram identificados através da análise do despacho nº18 de 2009 de S.Exª Almirante CEMA e aprovados em reunião no COMNAV a 21 de dezembro de 2009 (Deus, 2011). A informação contida neste despacho, combinada à experiência que a DAGI já possuía em virtude do desenvolvimento de outras ferramentas, permitiu assim definir um conjunto de requisitos operacionais exequíveis de serem implementados.

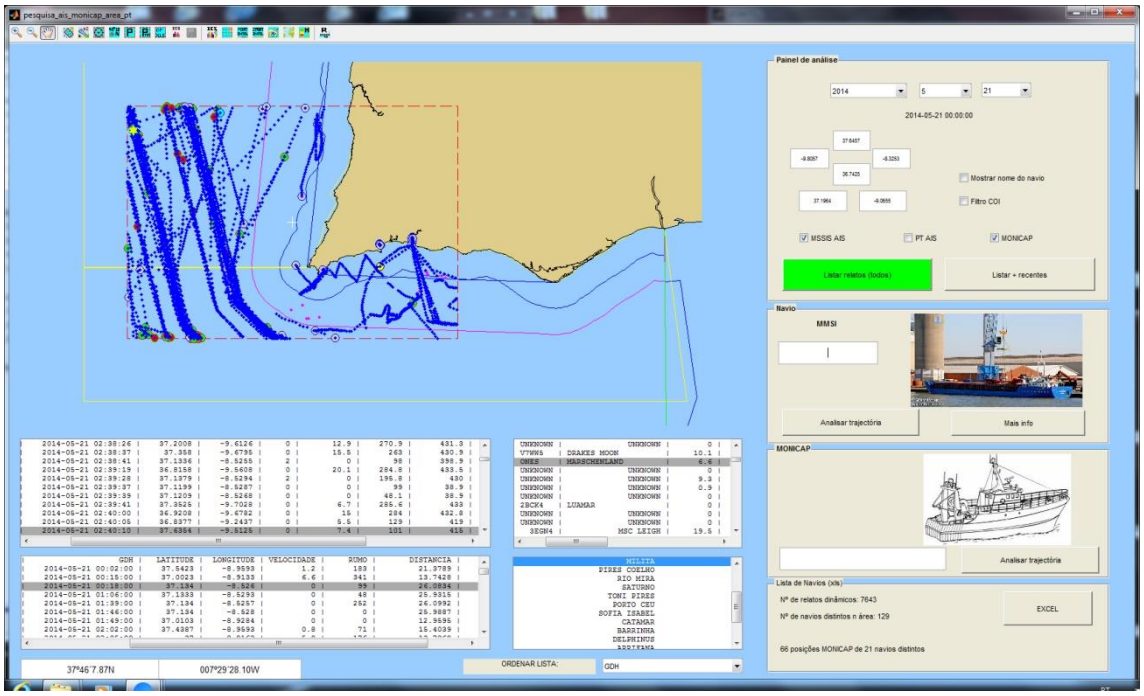


Figura 24 - Módulo de pesquisa em área.

Um dos módulos disponibilizados pela ferramenta AISINTEL é o módulo de pesquisa em área que permite visualizar navios equipados com o sistema AIS e MONICAP, calculando todos os indicadores de CSM[61] numa área e dia definidos pelo utilizador. Este módulo mostra ainda alguns elementos informativos (MMSI, posição geográfica, velocidade, rumo, estado de navegação, etc.) provenientes dos sistemas AIS e MONICAP, possibilitando a gravação destes elementos em XLS[62]. Este módulo possibilita a obtenção de mais informações acerca de um determinado navio através da interligação com o módulo base de dados de navios e sistema *Marine Traffic*. Este subsistema do AISINTEL disponibiliza também ao utilizador a interligação com o módulo de análise de trajetória, que será explicitado mais à frente. A informação que é disponibilizada ao utilizador está limitada pela memória RAM do computador e pela área desenhada pelo utilizador. Deste modo é conveniente que as áreas desenhadas não sejam muito grandes, devido ao excesso de relatos a serem carregados. Outra limitação deste módulo é o facto de o rumo representar a direção em graus no círculo trigonométrico e não a direção em relação ao norte geográfico.

Apesar de ter sido o primeiro módulo a ser desenvolvido, o módulo pesquisa em área tem sido objeto de constante evolução e atualizações em termos de interface

[61] Estes indicadores encontram-se definidos em MELO, L. C. (2010). *Indicadores de conhecimento situacional do espaço de envolvimento marítimo com recurso aos dados AIS*. Dissertação de mestrado em Ciências Militares Navais – Marinha na Escola Naval, Marinha.
[62] Formato do ficheiro utilizado pelo programa EXCEL.

gráfica. Uma das atualizações consistiu em disponibilizar os indicadores de CSM (Melo L.,2010) num botão na barra superior do interface principal deste módulo (antes os indicadores eram disponibilizados numa figura independente).

2.2.2 Módulo pares de navios

Outro módulo disponibilizado pelo protótipo AISINTEL é o módulo de análise de pares de navios. Este módulo permite visualizar pares de navios cuja sua posição tenha uma distância, em milhas náuticas, entre si igual ou inferior à definida pelo utilizador para uma determinada área também definida pelo utilizador. Os critérios para criar o "par" consistem não só na distância entre as duas posições, mas também na sua distância temporal. Para controlar a distância temporal este módulo disponibiliza uma lista de valores no painel "Nº de relatos a analisar". O valor por defeito são 10 relatos, ou seja, significa que os relatos (ordenados por GDH) serão comparados entre si em blocos de 10. Admite-se que os relatos em blocos diferentes possuem uma diferença substancial de GDH que não fará sentido analisar. Caso o utilizador selecione zonas que contenham portos, o número ocorrências será bastante maior, face aos navios atracados que se encontram a emitir sinal AIS.

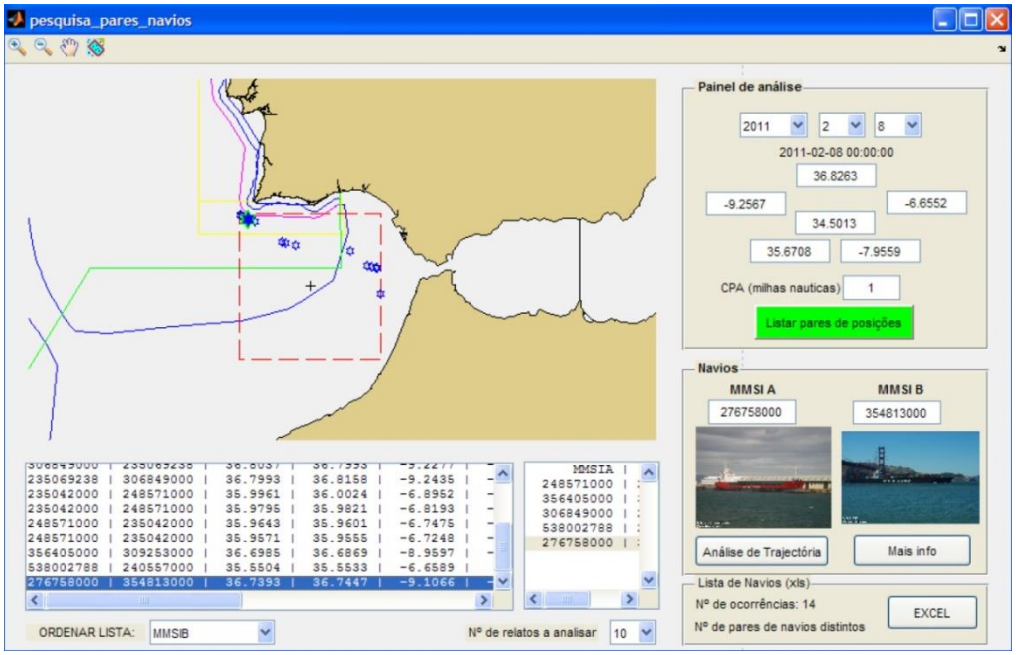


Figura 25 - Módulo pares de navios.

2.2.3 Módulo análise de trajetória e trajetórias simultâneas

Outra funcionalidade disponibilizada pelo AISINTEL é a análise de trajetória. Este módulo possibilita a reconstrução da trajetória de um navio durante um período de tempo definido pelo utilizador. Este módulo dá também a possibilidade de visualizar em modo "filme" o trânsito efetuado, pelo navio em análise, com indicação da posição geográfica, velocidade, rumo e condições meteorológicas. Para além do já mencionado, este módulo possui um gráfico com os perfis de variação de rumo e velocidade durante o trânsito realizado na janela temporal definida. Desta forma é possível identificar rapidamente os períodos durante a navegação, em que o navio ficou a pairar ou parou num porto. Apesar de este módulo permitir realizar todo o que acima foi referido, a informação disponibilizada está condicionada aos dados existentes, e estes por sua vez, estão limitados à qualidade da cobertura AIS. Desta forma, verifica-se que em algumas trajetórias carregadas, não existe 100% dos dados enviados pelos navios, o que provoca os espaços em branco como podemos verificar na Figura 26. Outra das limitações deste módulo é a de apenas tornar possível a visualização de uma única trajetória. Apesar das limitações descritas acima, este módulo representa uma das capacidades mais importantes disponibilizada nesta aplicação, pois permite a reconstrução dos "*vessel tracks*" e a sua análise *step-by-step*.

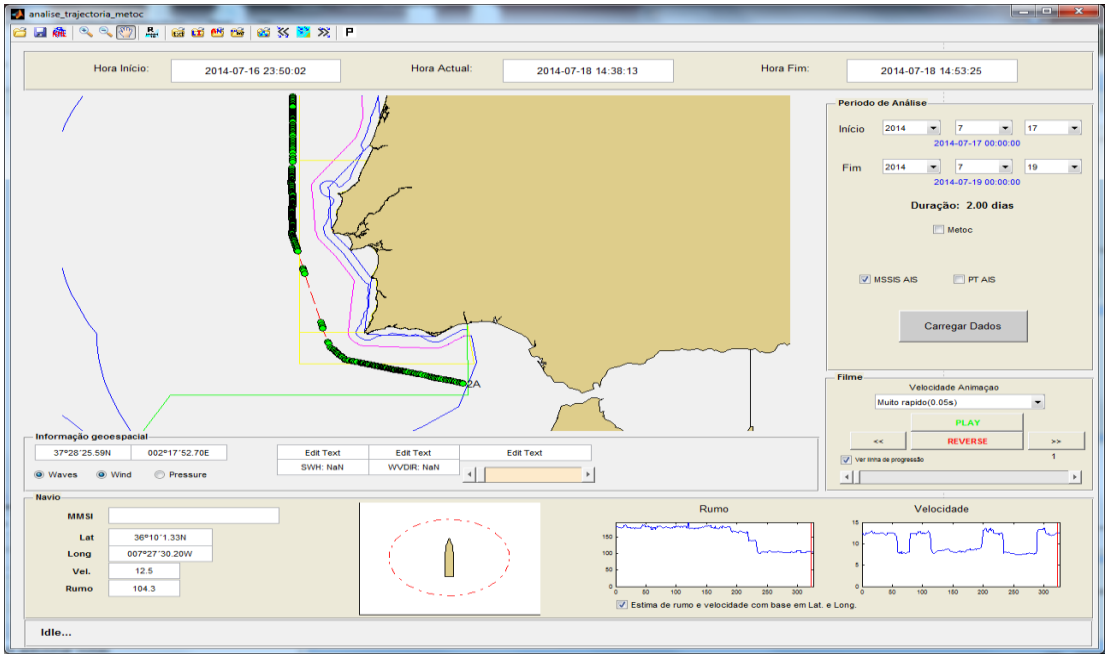
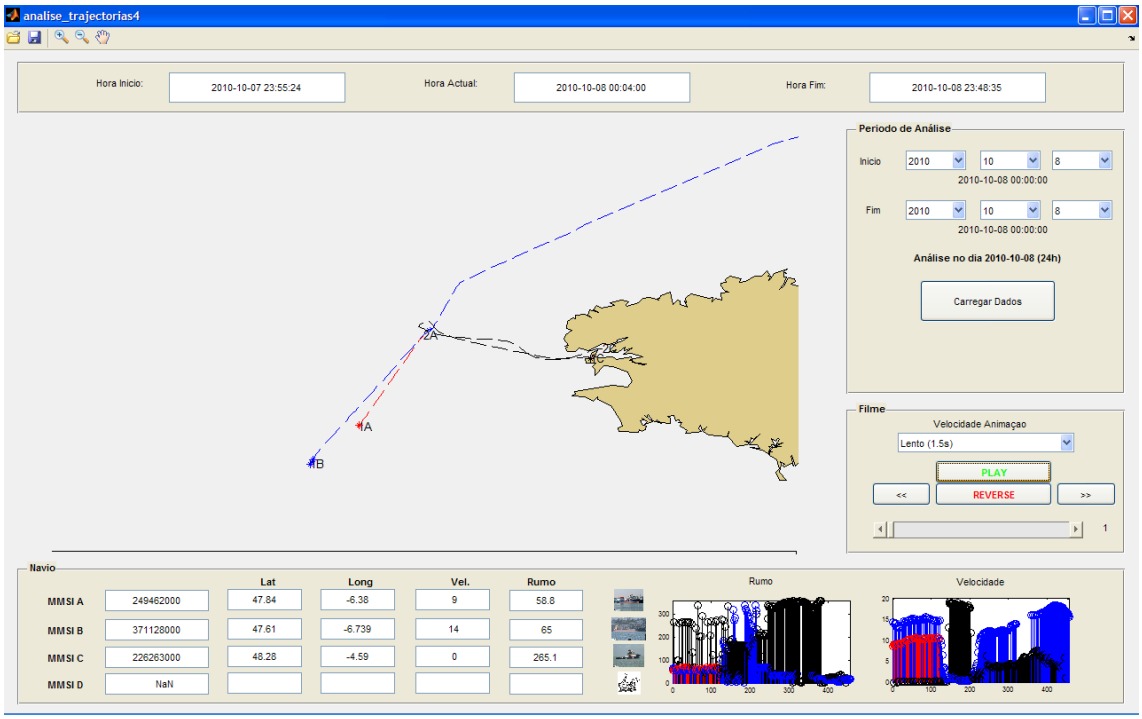


Figura 26 - Módulo de análise de trajetória.

Para além dos quatro módulos já referidos, o AISINTEL compreende ainda o módulo de análise de trajetórias simultâneas que é em tudo semelhante ao módulo de análise de trajetória. A principal diferença entre os dois módulos é o fato de o módulo de análise de trajetórias simultâneas possibilitar o acompanhamento de até quatro navios enquanto o módulo de análise de trajetórias permite apenas seguir um único navio. Outra diferença entre estes dois módulos é o fato de o módulo de análise de trajetórias simultâneas apenas mostrar no modo "filme" a posição geográfica, velocidade e rumo, não dando informação quanto às condições meteorológicas. Este módulo foi desenvolvido para visualizar mais do que uma trajetória de navios em simultâneo. O incidente que esteve na origem do desenvolvimento deste módulo ocorreu no dia 8 de outubro de 2010 onde dois navios colidem no Canal da Mancha.



2.2.4 Módulo análise de padrões

O módulo de análise de padrões tem como principal objetivo mostrar padrões de navegação, permitindo visualizar os navios equipados com o sistema AIS e MONICAP e a densidade de tráfego marítimo, numa área geográfica e período temporal definido pelo utilizador. Este módulo permite ao utilizador tratar os dados através da aplicação de outros filtros, para além dos inicialmente definidos (área e janela temporal), é ainda

possível aplicar filtros no que respeita à velocidade e rumo dos navios, possibilitando assim ao utilizador visualizar apenas os navios do seu interesse. O utilizador tem também a possibilidade de aplicar outros filtros através da função *query* personalizada.

Outra função que este módulo disponibiliza são os mapas de esforço, que possibilitam a identificação das principais rotas de navegação e a frequência com que são percorridas.

À semelhança do módulo anterior uma das principais limitações está relacionada com a memória RAM disponível, que limita a capacidade de carregamentos de dados para filtragem. Desta forma, para evitar sobrecarregar o processamento de dados e a memória disponível, optou-se por não fazer a correspondência dos pontos no mapa com a informação dos atributos AIS (MMSI, GDH, latitude, longitude, velocidade) na caixa de listagem. O carregamento apenas para visualizar densidades (sem usar os dados para filtragem) não possui limite de memória RAM, o que torna possível carregar um ano de dados.

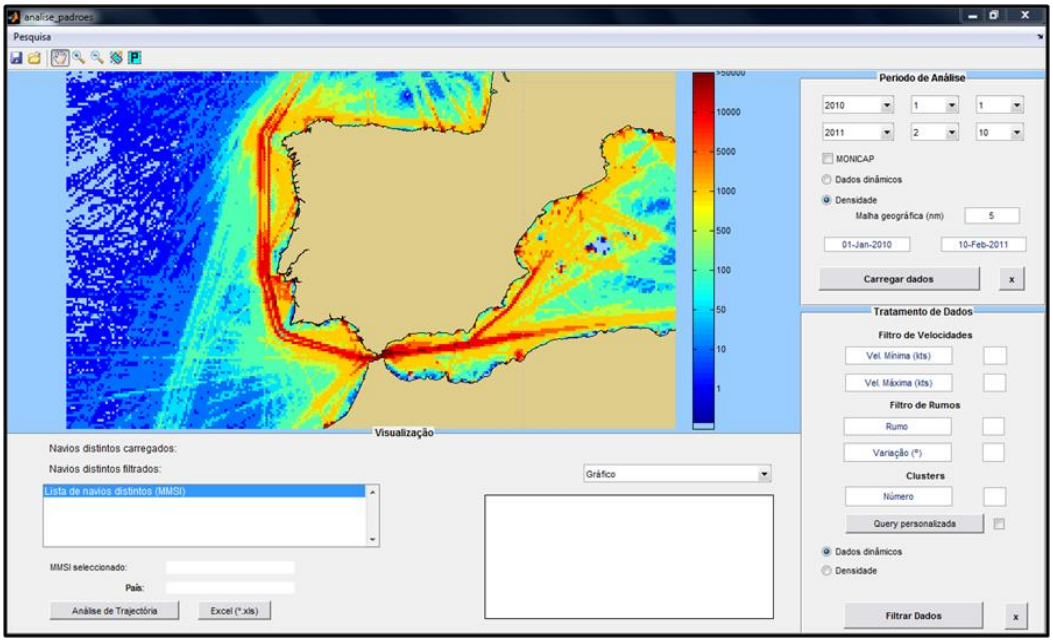


Figura 27 - Mapa de esforço do tráfego marítimo de navios equipados com AIS[63].

Este módulo implementa uma estrutura que permite realizar uma filtragem dos dados AIS e MONICAP, de modo a não sobrecarregar o sistema. Na fase inicial, a filtragem, é composta por dois filtros, um temporal e outro geográfico, permitindo assim carregar apenas os dados para a área e período que o utilizador deseja visualizar.

[63] Imagem retirada de GOMES, Carlos Amadeu Andrade (2012). *Modelos combinatórios para apoio ao planeamento de missão em operações de fiscalização e vigilância marítima*, Dissertação de mestrado em Ciências Militares Navais - Marinha na Escola Naval: Marinha, pp.15.

Posteriormente o utilizador poderá aplicar novos filtros (velocidade, rumo e *query* personalizada) de modo a visualizar apenas a informação desejada.

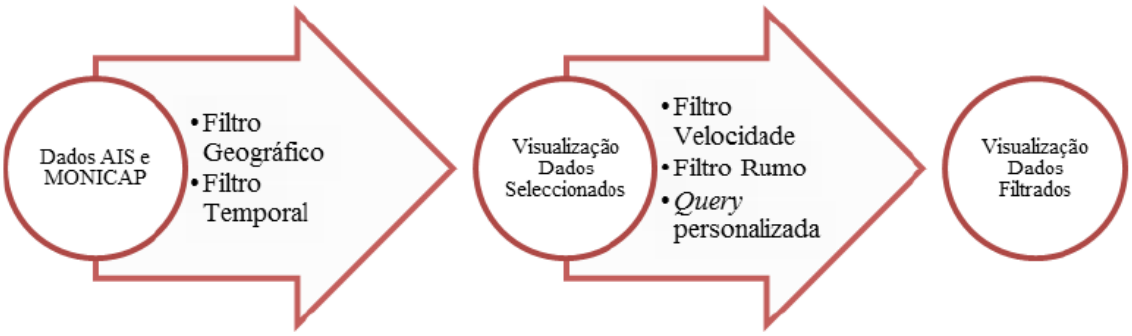


Figura 28 - Esquema estrutural do módulo análise de padrões[64].

Neste módulo é ainda possível identificar *clusters* de dados de forma a identificar zonas com padrão de comportamento semelhante (comportamento caracterizado por velocidade e rumo semelhantes).

2.2.5 Módulo base de dados de navios

Este módulo permite ao utilizador aceder a uma lista de todos os navios para os quais foi descodificado o respetivo MMSI. A base de dados associado a este módulo é composta por uma única tabela que relaciona um conjunto de atributos provenientes, não só de mensagens AIS de tipo 1, 2, 3, e 18 mas também de atributos descodificados de mensagens AIS de tipo 5 (informação estática). Desta forma este módulo permite ao utilizador consultar o nome, *callsign*, número IMO, MMSI, dimensões, tipo e último porto visitado do navio. As imagens associadas aos navios são registadas com o respetivo MMSI.

Uma das funcionalidades deste módulo confere ao utilizador a possibilidade de adicionar automaticamente novos navios à base de dados. Para tal, é necessário que o utilizador possua numa pasta os ficheiros com as mensagens AIS de tipo 5 (estáticas) descodificadas.

[64] Imagem retirada de MELO, Hugo Daniel Almeida de (2011). *Módulo de análise do quadro situacional marítimo para apoio a missões de vigilância e fiscalização marítima*, Dissertação de mestrado em Ciências Militares Navais - Marinha na Escola Naval: Marinha, pp.33.

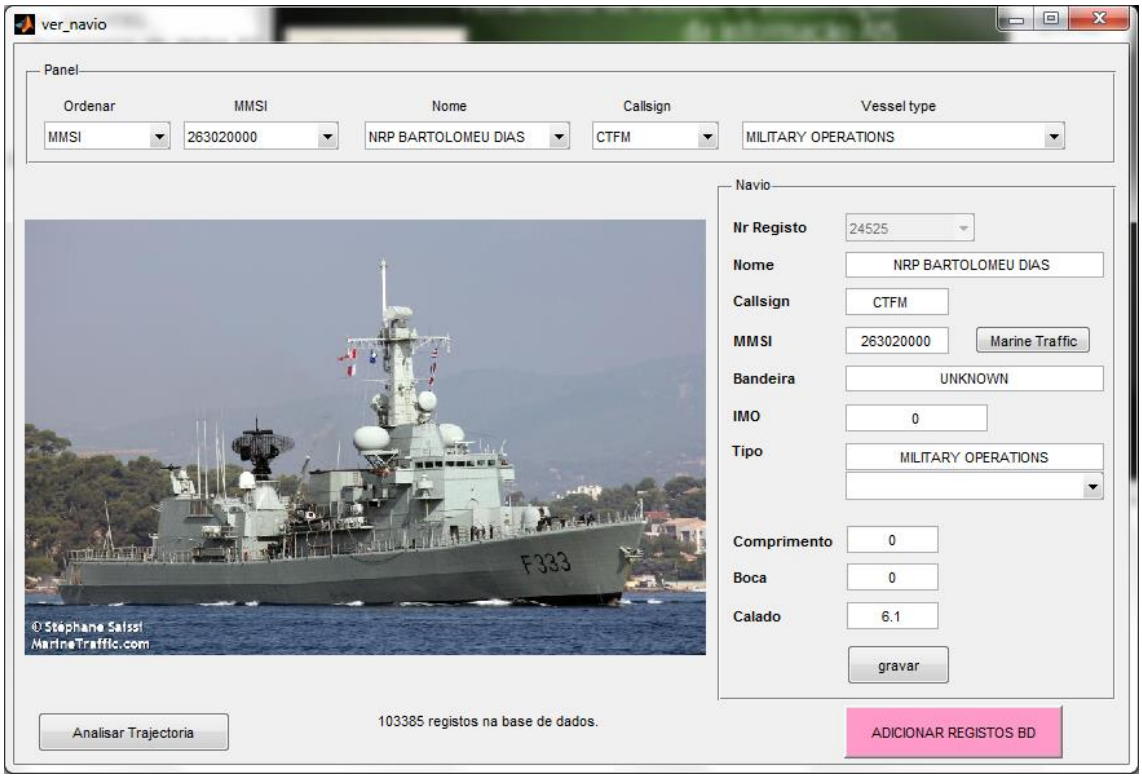


Figura 29 - NRP Bartolomeu Dias no módulo base de dados.

2.2.6 Módulo planeamento de patrulha

O módulo Planeamento de Patrulha foi desenvolvido por Gomes (2012) e permite a construção automática de um plano de patrulha que maximiza a co-localização de um meio naval (idealmente, um submarino) com a ocorrência no tempo e espaço de um determinado evento de interesse, sendo este último definido pelo utilizador.

O plano de patrulha obtido indica a subárea e o período horário em que o meio naval deverá efetuar a patrulha. Este plano de patrulha é solução de um problema de Programação Linear Binária e é mostrado no interface gráfico através de um diagrama de Gantt. Este problema encontra-se descrito em maior detalhe na dissertação de mestrado de Andrade Gomes (2012).

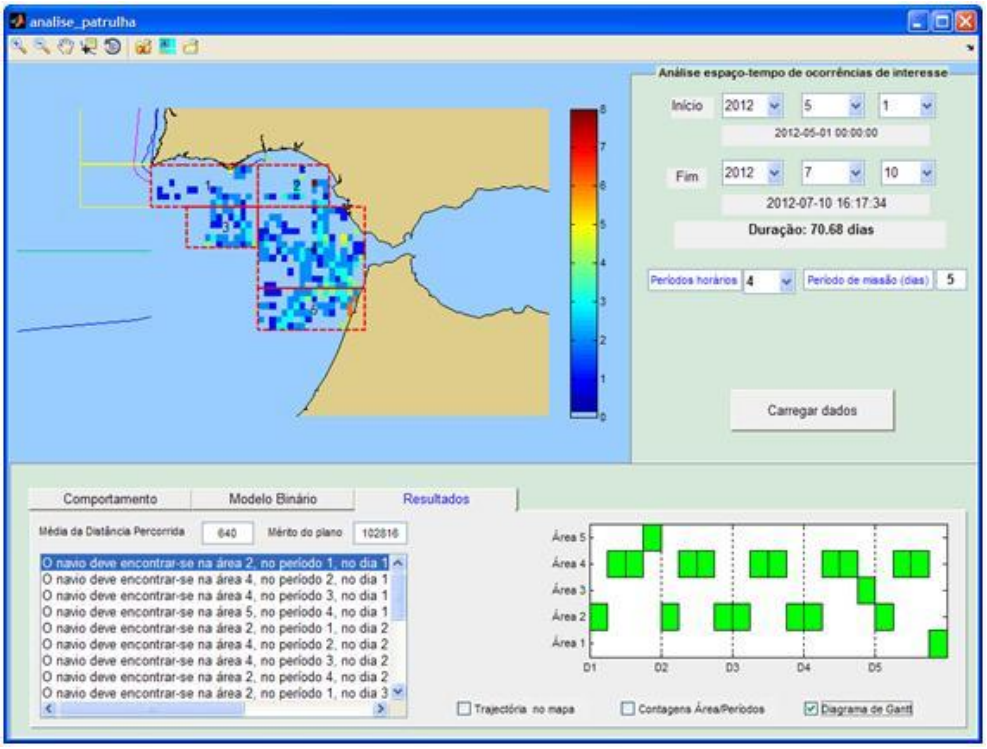


Figura 30 - Módulo planeamento de patrulha.

O planeamento de uma missão deverá indicar os períodos horários e áreas onde a unidade naval deverá efetuar a fiscalização ou vigilância marítima. Esta ferramenta propõe para cada dia de missão, as áreas e períodos horários onde a unidade naval deverá patrulhar de forma a maximizar a deteção de eventos suspeitos. Face a este tipo problema foram definidos os seguintes fatores que poderão ser impostos no planeamento de uma missão:

- Número de meios navais a utilizar.
- Cada área deve ser visitada pelo menos uma vez, durante toda a missão.
- Assegurar que durante o tempo da missão o navio se encontra sempre numa das áreas de patrulha.
- Construção de um planeamento onde se exige que todas as áreas sejam visitadas pelo menos um determinado número de vezes.
- Impossibilitar a patrulha consecutiva de duas áreas, em dois períodos horários consecutivos, quando a distância (em horas) entre essas duas áreas é superior a metade do período horário considerado no planeamento.

É de realçar que este módulo apenas aconselha o Comandante em que área e período horário o navio deverá estar, não condicionando a ação do Comandante da unidade naval na condução tática da operação naval.

2.2.7 Módulo AIS SAR

O Módulo AIS SAR permite ao utilizador calcular áreas de buscas para objetos que se encontrem à deriva, tendo em consideração as previsões meteorológicas no local do incidente, a hora do incidente e a hora estimada de chegada do meio de busca ao local. Estão disponíveis dois algoritmos para o cálculo das áreas de busca: área de busca segundo o método "*Totable Probable Error*" constante no manual NATO ATP-10 (D) e o método de estimação que recorre a simulação de Monte Carlo.

Os algoritmos que estão neste módulo foram inicialmente desenvolvidos para o SADAP. Contudo, em virtude de o AISINTEL estar desenvolvido numa versão mais recente do *software* MATLAB que o SADAP, este módulo permite o teste e experimentação de novas funções para o cálculo de áreas de busca e utilização de informação meteorológica. Uma das novidades que este módulo apresenta relativamente às semelhantes funcionalidades implementadas no SADAP consiste na visualização da altura significativa das ondas (*Significant Wave Height*) através de um mapa de temperatura.

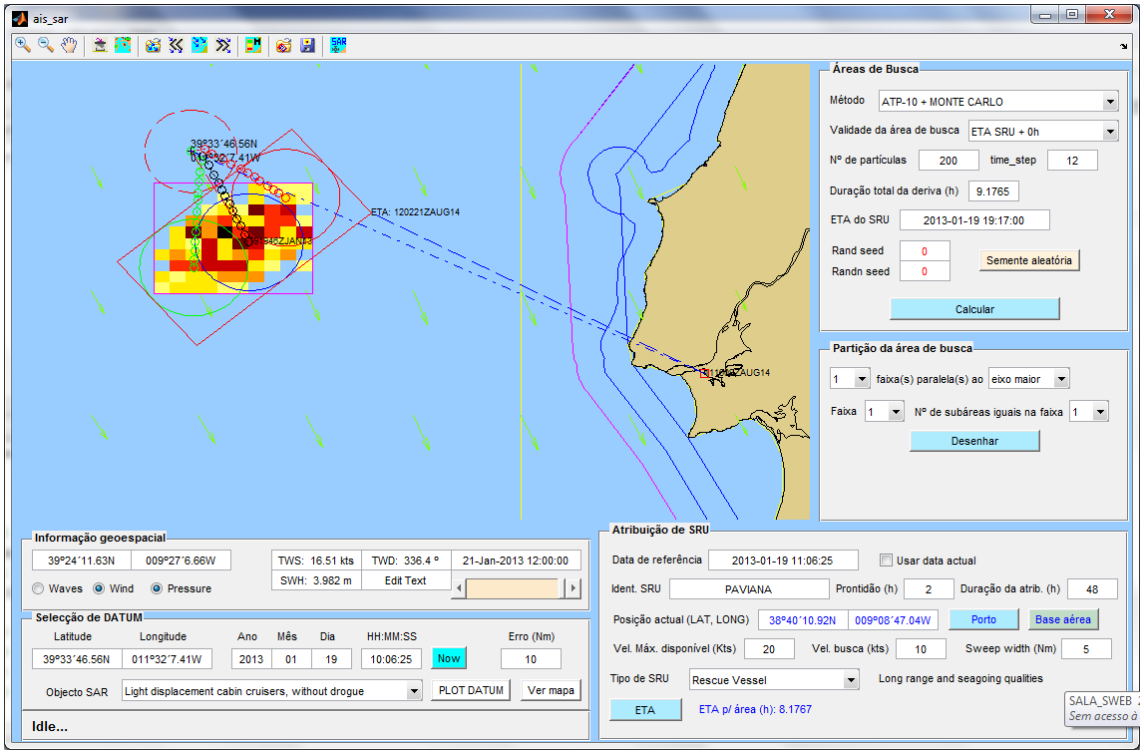


Figura 31 –Módulo AIS SAR

2.2.8 Módulo análise de incidentes

No início de 2013, a DAGI foi contactada pelo Gabinete de Prevenção e de Investigação de Acidentes Marítimos (GPIAM) no sentido de auxiliar a investigação que estava a decorrer relativo ao encalhe do navio "MERLE". Este navio encalhou em 19 de janeiro de 2013 na praia da Torreira, perto de Aveiro. As condições meteorológicas que se fizeram sentir nesse dia eram bastante adversas, tendo sido registada pela boia ondógrafo de Leixões ondas de 16 metros no período que antecedeu o encalhe do navio.

O contacto com a DAGI foi efetuado pelo CMG Velho Gouveia, em que foi solicitado que se fizesse o cruzamento das posições do navio com a informação áudio e meteorológica disponível. O desafio consistiu em integrar diferentes fontes de informação (AIS, METOC, Áudio, texto) num único interface. Este interface deve permitir o carregamento destas fontes de dados, a sua sincronização e visualização em modo "filme". O trabalho realizado teve como base o módulo de análise de trajetória, onde foram adicionados novos objetos e funções para integrar, sincronizar e visualizar as novas fontes de dados disponíveis. Como resultado final, obteve-se o módulo de análise de incidentes.

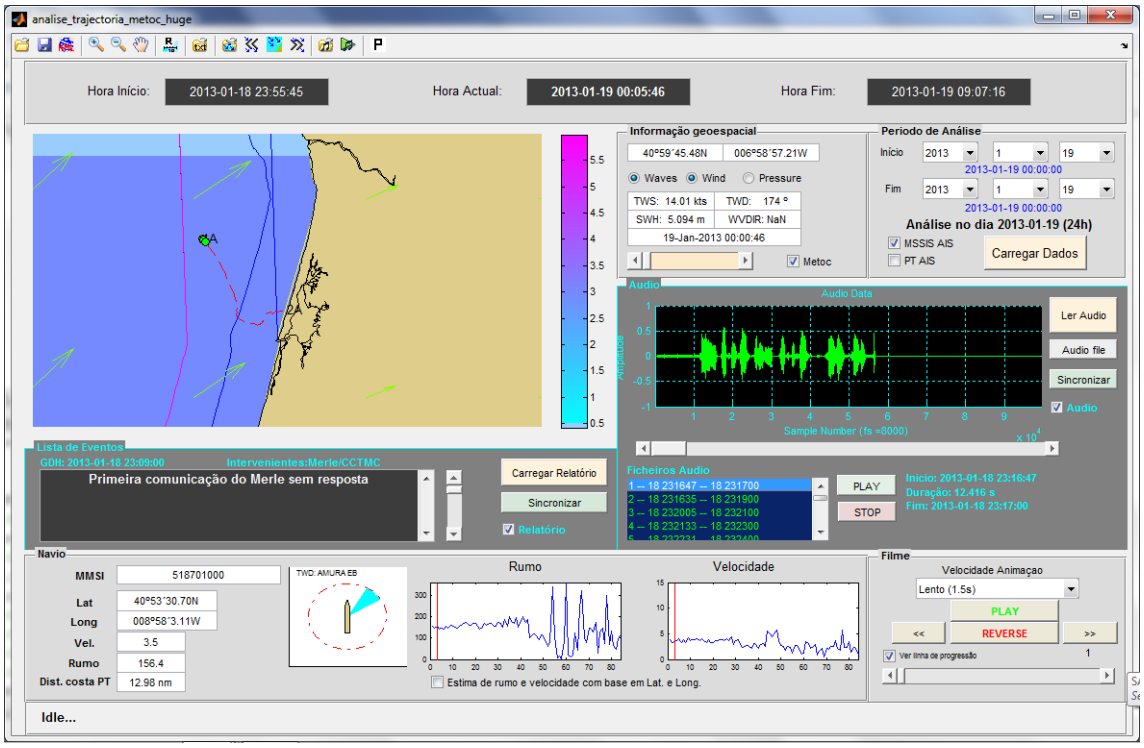


Figura 32 - Incidente com a embarcação Merle.

O trabalho realizado permitiu analisar as condições meteorológicas durante o trajeto do navio até ao momento em pretendia entrar na barra de Aveiro e cruzar esta informação, não só com as decisões do Comandante, mas também com as orientações dadas pelas autoridades nacionais.

Após a realização deste trabalho, o módulo de análise de incidentes foi apresentado ao Secretário de Estado do Mar em 9 de maio de 2013.

2.3 TRITON – futuro sistema de CSM NATO

É um fato que a NATO possui alguns sistemas como o MCCIS que já se encontram bastante desatualizados, apesar das sucessivas modernizações efetuadas. Neste sentido a NATO tem vindo a desenvolver esforços no intuito de melhorar os seus sistemas de CSM. Para isso, foi criado um grupo de trabalho com o objetivo de desenvolver o novo sistema NATO de CSM, designado por TRITON.

O objetivo deste projeto é oferecer capacidades de comando e controlo (C2) a todos os centros de comando da estrutura NATO. Este sistema deve ser capaz de operar em diferentes cenários como: tempo de paz, resposta a crises, exercícios e operações de contingência, servindo sempre os interesses nacionais[65].

A implementação de funcionalidades, neste projeto, ocorrerá de forma faseada, sendo que na primeira fase este sistema conterá uma nova versão do MCCIS e funcionalidades de CSM. A segunda e terceira fase incorporaram, respetivamente, funcionalidades de guerra de minas e guerra anfíbia. Atualmente a primeira fase já se encontra concluída e em fase de testes, não havendo ainda data para a conclusão das fases seguintes.

O sistema TRITON só atinge a plenitude da capacidade operacional quando todas as fases forem concluídas. No entanto, a capacidade operacional inicial é alcançada com a conclusão da primeira fase do projeto[65]. A execução da primeira fase do projeto TRITON serve também para efetuar a transição do sistema MCCIS para o sistema TRITON, sendo garantido que não há perdas de funcionalidades durante este período.

Como o TRITON pode ser o núcleo de capacidades marítimas de C2 de uma nação, a arquitetura deste sistema é desenhada de forma a ser adaptável e flexível, permitindo que as funcionalidades específicas nacionais sejam interligadas a este sistema. A arquitetura do TRITON e os produtos por ele fornecidos devem ser geridos por cada

[65] NATO (2011). *Project 0IS0308100 – Provide functional services for command control of maritime operations*. Norfolk, NATO.

nação de modo a permitir a verificação do sistema, otimização do uso da infraestrutura e modalidades de apoio.

As funcionalidades de CSM, desenvolvidas para o projeto TRITON são atualmente utilizadas interinamente pelo CCMAR Nápoles e CCMAR Northwood em apoio a operações como: *Active Endeavour*[66], *Ocean Shield*[67] e *Unified Protector*[68]. Recentemente na operação *Unified Protector*, a NATO verificou que uma infraestrutura de informações de capacidade C2 é uma condição essencial para o sucesso das operações combinadas[69]. Desta forma, as nações podem contar com a coesão e interoperabilidade, entre os níveis estratégico, operacional e tático, permitindo a condução de operações conjuntas[70] e combinadas.

A interoperabilidade é um dos principais desafios para o TRITON. Neste sentido, este protótipo será submetido a testes de interoperabilidade no NATO *Coalition Warrior Interoperability Exercise*. A tecnologia na área de serviços de informação evolui muito rapidamente, no entanto, é expectável que a vida útil do projeto TRITON seja de cerca de 20 anos.

À data da escrita desta dissertação, a empresa EDISOFT participa no grupo de trabalho de desenvolvimento do TRITON, constituindo a principal fonte do que é dito nesta secção.

2.4 Métricas para desenvolvimento de software

As métricas permitem avaliar e validar um projeto ou tarefa, sendo a sua definição, construção e aplicação uma fase importante no desenvolvimento de um projeto. Para Sommerville (2011) a métrica de um *software* é uma característica de um programa que pode ser objetivamente medida[71]. A utilização de métricas de avaliação permite a validação, medição do desempenho e identificação das fragilidades do sistema. Deste

[66] A operação *Active Endeavour* "é uma operação marítima da NATO no Mediterrâneo Oriental, como parte da campanha internacional contra o terrorismo. A sua missão é conduzir operações marítimas na área de operações atribuídas e demonstrar a determinação da NATO para ajudar a dissuadir, defender, impedir e proteger contra o terrorismo" (EMGFA (s.d). *NATO OAE (Operation Active Endeavour)*. Retirado de http://www.emgfa.pt/pt/operacoes/missoes/oae-mediterraneo, consultado a 26 de julho de 2014).
[67] Operação *Ocean Shield* "é a contribuição da NATO para os esforços internacionais para combater a pirataria ao largo do Corno de África, iniciada 17 Agosto 2009" (EMGFA (s.d.). *NATO Ocean Shield*. Retirado de http://www.emgfa.pt/pt/operacoes/missoes/oceanshield, consultado a 26 de julho de 2014).
[68] O objetivo da operação *Unified Protector* era proteger civis e áreas povoadas por civis sob ataque ou ameaça de ataque. Esta operação decorreu na Líbia entre 31 de março de 2011 e 31 de outubro de 2011.
[69] Termo utilizado para operações realizadas entre as forças armadas de diferentes nações.
[70] Termo utilizado para operações realizadas entre os diversos ramos das forças armadas de uma nação.
[71] SOMMERVILLE, Ian (2011). *Software Engineering* (9th ed.), Boston, Addison-Wesley, pp.668.

modo é assim possível aperfeiçoar o *software* desenvolvido através da identificação, medição e controlo dos principais parâmetros do programa. Após a aplicação das métricas pode-se então tirar ilações no que diz respeito ao desempenho da ferramenta e alcance do objetivo.

As métricas podem ser divididas em métricas diretas e indiretas. As primeiras dizem respeito a uma medida realizada em termos de atributos observados, enquanto que as métricas indiretas referem-se a medidas obtidas a partir de outras métricas (Vasconcelos A., 2005). Desta forma, o autor decidiu aplicar métricas diretas, de modo a testar e validar o interface de monitorização de alertas, para o efeito foram criados três alertas como iremos ver mais à frente na secção 4.1.

No capítulo 3 são apresentadas um conjunto de métricas para avaliar a eficiência das rotinas que avaliam os atributos de um alerta e também métricas que caracterizam a ocorrência do evento de interesse associado ao alerta.

Capítulo 3

Modelação de Alertas

3.1 Especificação de alertas

3.2 CADOP – requisitos técnicos e operacionais

3.3 Implementação do módulo de alertas

3.4 Métricas para avaliação de alertas

3 Capítulo 3 – Modelação de Alertas

Como referido no capítulo anterior os sistemas de informação na área do CSM constituem uma mais-valia para o esclarecimento do panorama de superfície, permitindo a identificação de casos de perigo. Estas aplicações permitem, ao seu utilizador, tomar uma decisão sustentada, possibilitando um melhor e mais atempado emprego dos meios. Um dos requisitos identificados pela comunidade operacional, consiste na monitorização de navios que se encontram indiciados da prática de atividades ilegais ou em risco de segurança. Pesquisar situações desta natureza exige um esforço considerável aos operadores dos sistemas de informação de CSM, pois requer que o operador detenha um conjunto de informações que nem sempre as tem ou estão disponíveis, ou exige um nível de conhecimento avançado no uso destes sistemas que, também, nem sempre é o caso. Por este motivo entende-se que é pertinente desenvolver um conjunto de funcionalidades, em si, muito específicas, mas bastante direcionadas para a realização de uma tarefa em concreto que se resume à construção de uma alarme e respetiva monitorização. O desafio inerente ao presente trabalho incide na construção de um instrumento que seja flexível e de fácil utilização pelos operadores para a monitorização de eventos de interesse. Pretende-se que o módulo de Alertas apresente um equilíbrio entre flexibilidade na parametrização do alerta e na facilidade de uso pelos operadores.

Neste capítulo pretende-se definir sucintamente o conceito de alerta, explicitando os principais parâmetros que o constituem. Posteriormente serão identificados os principais requisitos operacionais definidos pelo CADOP, assim como os alertas mais relevantes para a costa portuguesa. De seguida descreve-se os algoritmos de pesquisa associados aos alertas e os interfaces desenhados para a construção e monitorização destes. No final serão ainda apresentadas as métricas mais relevantes para avaliar um alerta em termos de performance. Neste trabalho foi utilizado o *software* MATLAB da *Mathworks* para a implementação de algoritmos e interface gráficos.

3.1 Especificação de alertas

Monitorizar eventos de interesse no espaço marítimo implica a construção e monitorização de alertas. O conceito de alerta apresenta diversas definições, sendo definido como "*sinal para se estar alerta vigilante para o ataque, defesa ou proteção*"[72], é também sinónimo de "*vigilante; sobreaviso*"[72]. A Proteção Civil portuguesa define alerta como a "*comunicação de uma emergência feita a qualquer dos órgãos operacionais do sistema de proteção civil, por um indivíduo ou entidade, devendo ser acompanhada dos elementos de informação essenciais a um conhecimento perfeito da situação*"[73].

Importa também definir o que é um sistema de alerta. Para Macedo e Sardinha "*um sistema de alerta permite desencadear automaticamente, muito antes da efetivação do risco, em cada região e momento, um conjunto de medidas e ações tendentes a evitar ou mitigar a ocorrência*"[74]. Não esqueçamos que "*um alerta bem difundido, coerente e credível vem permitir aos agentes aumentar o seu nível de prontidão associado a um pré-posicionamento de meios*"[75], permitindo responder aos alarmes de forma mais eficaz e eficiente.

Na fase inicial deste trabalho, a investigação centrou-se principalmente na análise das dimensões que constituem um alerta perante os requisitos operacionais definidos e os dados AIS disponíveis. Uma dimensão representa uma estrutura de dados que possibilita orientar o utilizador a analisar um fato sob uma determinada perspetiva.

Face à grande quantidade de variáveis que podem ser consideradas quando se trabalha com dados provenientes do meio marítimo, foram identificadas as seguintes dimensões, que se consideraram relevantes para a construção de um alerta:

[72] Alerta in Infopédia. Porto Editora (2014). *Infopédia*. Retirado de http://www.infopedia.pt/lingua-portuguesa/alerta, consultado em 22 de março de 2014.
[73] PROTEÇÃO CIVIL (s.d.). *Normas para conceção do sistema de alerta e aviso no âmbito dos planos de emergência interna de barragens*, pp.5, consultado em 22 de março de 2014, http://www.proteccaocivil.pt/RiscosVulnerabilidades/RiscosNaturais/SegurancaBarragens/Documents/NORMAS%20Barragens.pdf
[74] MACEDO, F. Wolfango de & SARDINHA, A.M. (1987). *Fogos Florestais*, 2º Volume, Publicações Ciência e Vida, Lisboa, pp342.
[75] ACEL (1989). *Manual de Prevenção e Luta contra os Incêndios Florestais*, traduzido do manual com o mesmo nome elaborado pelo ICONA, Lisboa, pp49.

Tabela 2 - Dimensões do objeto "alerta".

Dimensão	Descrição
Área	É a dimensão mais importante pois qualquer fato ou evento de interesse ocorre num determinado lugar ou posição geográfica.
Tempo	Define a data de validade para o alerta e o período do dia em que o alerta é válido para ser pesquisado.
Navio	Esta dimensão permite caracterizar o objeto "Navio". O utilizador pode selecionar a pesquisa de um único tipo de navio, ou de uma lista de navios agregada por uma ou mais combinações de características comuns. Por exemplo: pretende-se detetar todos os navios de tipo *tanker* que entrem na Área Operacional da Marinha (AOM) com velocidade inferior a 5 nós.
Metoc	Esta dimensão agrega um conjunto de atributos relacionados com grandezas físicas disponibilizadas nos ficheiros Grib[76]. Estes atributos correspondem à direção e velocidade do vento, pressão atmosférica e altura significativa das ondas.
Dados do alerta	Esta dimensão agrega um conjunto de atributos onde o utilizador pode guardar o nome do alerta, a sua identificação, a data de criação, a prioridade e alguns comentários sobre o alerta.

3.2 CADOP – requisitos técnicos e operacionais

Nesta secção pretende-se identificar os principais requisitos que estiveram na base do desenvolvimento do módulo de alertas. Para o efeito foi contactado o CADOP, na figura do atual Diretor Interino, o CFR Cavaleiro Ângelo, orientador desta dissertação, com o intuito de identificar e caracterizar um conjunto de requisitos operacionais para situações de risco e padrões suspeitos mais relevantes na costa portuguesa, de modo a produzir alertas credíveis e adequados. Neste sentido foram identificados e adotados para este trabalho os seguintes **requisitos operacionais**, que de uma forma geral, correspondem a eventos (ou factos) de interesse:

i. Navios que se aproximem de costa e tenham rumo na direção da mesma.
ii. Interdição de navios que entrem em áreas de interesse[77].
iii. Identificação de contatos de interesse que entrem na AOM.

[76] Os ficheiros GRIB são um "*formato de dados matematicamente conciso e bastante utilizado na meteorologia como forma de armazenar dados meteorológicos e de previsão*" (Brito B.,2011).
[77] Áreas de interesse são áreas onde se realizem exercícios como por exemplo exercícios com submarinos ou exercícios de tiro.

iv. Navios que entrem na zona de separação de tráfego.

v. Navios com velocidades inferiores a 5 nós.

Como requisitos do interface gráfico, foram definidos os seguintes **requisitos técnicos**:

i. A aplicação para a criação de alertas deve permitir a parametrização de um alerta onde o utilizador possa escolher a área onde ocorre o evento de interesse, eventual período horário associado ao evento e altura significativa das ondas associado ao evento

ii. A aplicação deverá permitir guardar o alerta criado e alterá-lo posteriormente.

iii. A aplicação deverá disponibilizar dois interfaces independentes para a construção e monitorização de alertas.

iv. Ambos os interfaces devem disponibilizar um mapa para visualizar a área associada ao evento de interesse.

v. A aplicação deve fornecer a identificação dos navios, através do MMSI.

3.3 Implementação do módulo de alertas

O módulo de alertas é constituído por dois interfaces gráficos principais. O primeiro interface gráfico permite criar e guardar a informação inerente a um alerta. O segundo interface gráfico permite monitorizar um ou vários alertas. A implementação de rotinas e dos interfaces gráficos é feito com recurso ao *software* MATLAB. Ao longo dos anos o MATLAB tem sido o instrumento preferencial para a implementação de rotinas em diversos projetos desenvolvidos pela DAGI, donde se inclui o protótipo AISINTEL. A utilização desta ferramenta tem inúmeras vantagens que "*passam pela acentuada curva de aprendizagem, pela reutilização de código feito em trabalhos anteriores e pela utilização de estruturas de dados já em uso pelo AISINTEL*"[78] (Gomes, 2012).

A implementação do interface de construção de um alerta apenas requereu a programação necessária para associar os objetos da tela do interface (*pushbutton*,

[78] GOMES, C. A. (2012). *Modelos combinatórios para o apoio ao planeamento de missão em operações de fiscalização e vigilância marítima.* Dissertação de mestrado em Ciências Militares Navais - Ramo Marinha, Escola Naval, Marinha, pp42.

togglebutton, axes[79], etc) com as estruturas de dados que definem o alerta. A implementação do interface de monitorização requereu o mesmo tipo de programação referido para o interface de construção de alertas, mas acrescido de rotinas e algoritmos específicos para avaliar as condições lógicas inerentes ao alerta e também rotinas para visualização dos mesmos. Antes de se implementar ambos os interfaces é necessário implementar a estrutura de dados que representa um alerta. Esta estrutura de dados será alterada e servirá de *input* em ambos os interfaces.

3.3.1 Estrutura de dados do objeto alarme

Para melhor entender a estrutura de dados definida para um alerta, é necessário revisitar a estrutura dos dados AIS que estão disponíveis para análise. Neste sentido, será apresentado, de forma sucinta, um resumo de todo o processo, de forma a ser percetível ao leitor comum.

Como já foi referido anteriormente os dados AIS são disponibilizados pelo CCMAR Northwood, sendo gravados no servidor AIS sediado na DITIC. "*Esta informação é sucessivamente incrementada, sendo criado então um ficheiro do tipo *.mat, formato passível de ser carregado e tratado em MATLAB, com o correspondente a 10 minutos de dados AIS "em bruto". Estes ficheiros são então descodificados, recorrendo às rotinas desenvolvidas no âmbito da aplicação SADAP (desenvolvida pelo CMG Maia Martins), e compilados num ficheiro diário, denominado ficheiro do tipo "F" (de footprint)"*[80]. Cada ficheiro do tipo "F" é composto por duas sub-matrizes: a sub-matriz que contém dados dentro da AOM (*footprint_aom.mat*) e a sub-matriz com dados fora da AOM (*footprint_aom.mat*). Estas rotinas iniciaram-se em janeiro de 2010, contando até à data com cerca de 99,2 GB de dados AIS. Importa salientar que são estes ficheiros que alimentam a aplicação AISINTEL e o módulo desenvolvido nesta dissertação de mestrado.

[79] Ferramenta existente no MATLAB que permite a criar de gráficos.
[80] MELO, Hugo Daniel Almeida de (2011). *Módulo de análise do quadro situacional marítimo para apoio a missões de vigilância e fiscalização marítima*, Dissertação de mestrado em Ciências Militares Navais - Marinha na Escola Naval: Marinha, pp.30.

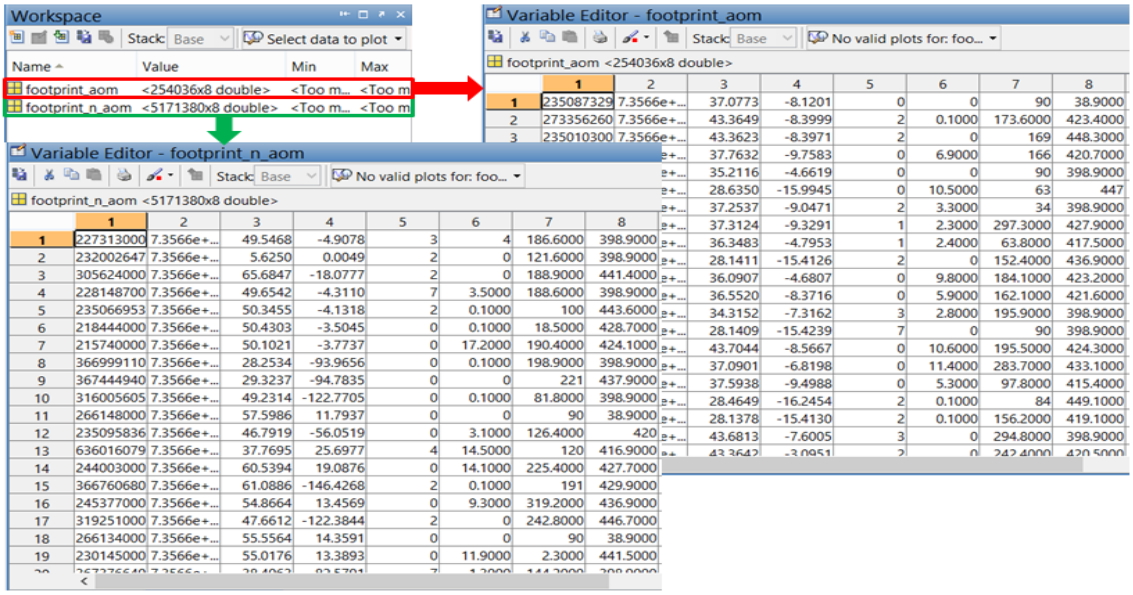


Figura 33 - Sub-matrizes que compõem um ficheiro diário de dados AIS.

Outro fator importante a compreender, é a necessidade de criar filtros temporais e geográficos. Estes filtros compõem a filtragem inicial dos dados AIS e foram programados de forma integrada, pois os dados encontram-se organizados sob a forma de ficheiros diários, isto é, apenas após ser definido a janela temporal e área é que a aplicação carrega os dados.

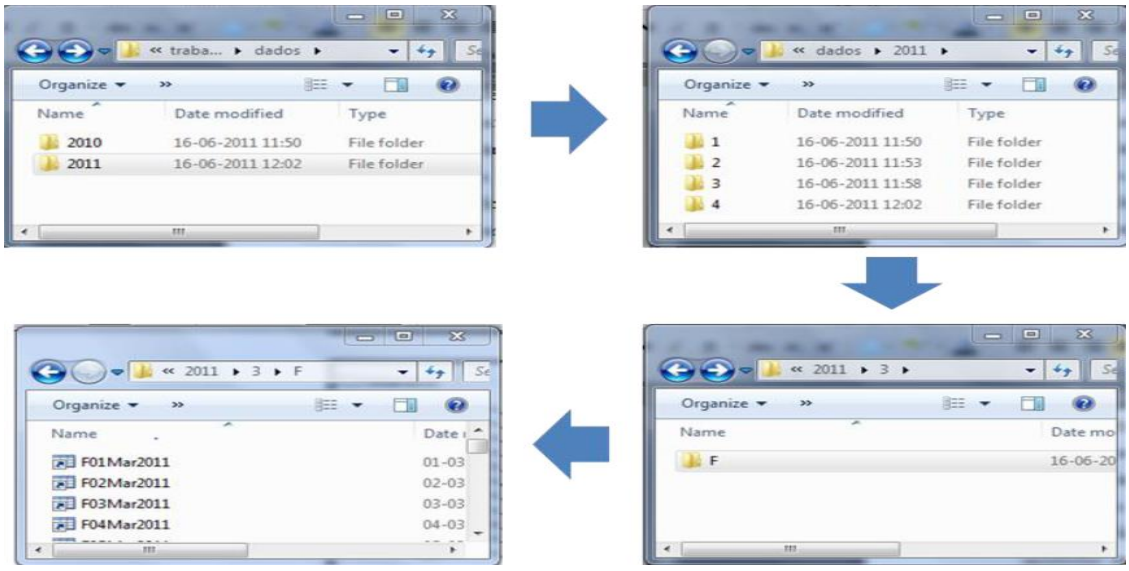


Figura 34 - Organização dos dados AIS em ficheiros diários[81].

É também de salientar que a explicação detalhada de todo este processo de tratamento e descodificação de dados, não será abordado neste trabalho, por já se

[81] Imagem retirada de MELO, Hugo Daniel Almeida de (2011). *Módulo de análise do quadro situacional marítimo para apoio a missões de vigilância e fiscalização marítima*, Dissertação de mestrado em Ciências Militares Navais - Marinha na Escola Naval: Marinha, pp.31.

encontrar perfeitamente explicitado em outros trabalhos de investigação, disponíveis para consulta pública (Melo L., 2010; Melo H., 2011).

No caso do alerta, a organização dos atributos que o compõem é um pouco diferente, visto que os dados se encontram agrupados numa *struct*[82] e não numa matriz. A *struct* de cada alarme é composta por vários campos (*fields*), que por sua vez, correspondem a *cell arrays*[83], *strings*[84] e *values*[85], que representam os parâmetros anteriormente mencionados. Os *cell arrays* são usados para guardar os parâmetros da área, tempo, navios e metoc, enquanto as *strings* e *values* são usados para guardar os dados do alarme como o nome, utilizador, data, comentários e prioridade.

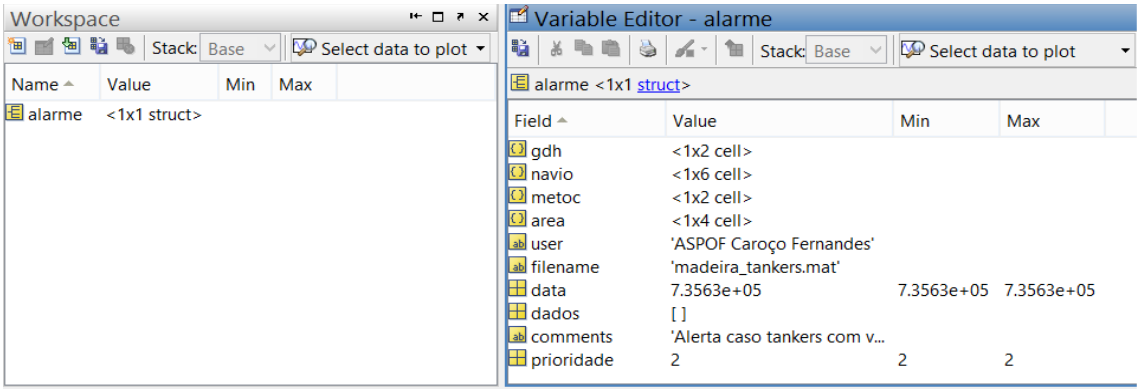


Figura 35 - Estrutura de dados de um alerta previamente construído no módulo de alertas.

Ao carregar o ficheiro "madeira_tankers.mat" na linha de comandos do MATLAB é carregado em memória a *struct* "alarme". Esta estrutura pode ser visualizada executando a instrução com o mesmo nome na linha de comandos do MATLAB:

```
alarme =

          gdh: {[0]  {2x3 cell}}
        navio: {[1]  [1]  []  []  [1 15 20]  [0 20 0]}
        metoc: {[1]  {4x4 cell}}
         area: {[1]  {1x6 cell}  [1]  [1]}
         user: 'ASPOF Caroço Fernandes'
     filename: 'madeira_tankers.mat'
         data: 7.3563e+005
        dados: []
     comments: 'Alerta caso tankers com velocidads entre os 15 e 20 nós e mar entre 7 e 15m se encontrem nas águas do arquipélago da Madeira. '
   prioridade: 2
```

Figura 36. Visualização do conteúdo dos *fields* da *struct* alarme.

[82] Ferramenta do MATLAB que permite agrupar todo o tipo de dados (*cell array*, *string*, entre outros) numa só estrutura.
[83] Um *cell array* é um conjunto de várias células, em que cada uma delas pode conter qualquer tipo de dados, como: matrizes, texto, número e combinações de texto e números, entre outros.
[84] Formato de dados do MATLAB que permite guardar texto.
[85] Formato de dados do MATLAB que permite guardar números.

- Field “gdh”

O campo “gdh” corresponde a um *cell array* de dimensão 1x2 (uma linha e duas colunas). A componente (1,1) deste *cell array* corresponde a uma variável binária que indica se o alerta incide num período horário específico pelo utilizador, ou não, tendo a seguinte representação:

$$alarme.gdh\{1,1\} = \begin{cases} 1 & se\ estiver\ selecionada\ a\ opção\ Incluir\ condição\ lógica \\ 0 & caso\ contrário \end{cases}$$

A componente (1,2) corresponde a um *cell array* de dimensão 2x3, em que a primeira coluna tem a descrição das componentes seguintes (2ª e 3ª colunas). As componentes (1,2) e (1,3) guardam, respetivamente, a data de validade inicial e final do alerta no formato *datenum*[86], enquanto que as componentes (2,2) e (2,3) guardam, respetivamente, o período horário inicial e final do alerta no formato *string*.

```
>> alarme.gdh{1,2}

ans =

    'validade'          [7.3563e+005]    [7.3566e+005]
    'periodo horário'   '10'             '12'
```

Figura 37 - Componente (1,2) do campo "gdh".

Por exemplo, no alerta “madeira_tankers.mat” pretende-se alertar o utilizador quando navios de tipo “tanker” entrem na ZEE da Madeira entre as 10:00 e as 12:00.

- Field “navio”

O campo “navio” corresponde a um *cell array* de dimensão 1x6. Neste campo a componente (1,1), à semelhança da componente (1,1) do campo anterior, corresponde a uma variável binária que representa se o alerta tem associado uma lista de navios ou tipo de navio. Caso esta variável seja nula então o alerta poderá incidir exclusivamente nas condições meteorológicas da área selecionada[87]. A componente (1,2) reproduz a mesma variável binária, tendo a seguinte representação:

[86] É uma função do MATLAB que permite converter uma data num único número. Por exemplo, datenum('13-Aug-2014') = 735824.
[87] O utilizador pode construir um alerta para ser avisado quando as condições meteorológicas atinjam um determinado nível numa área selecionada. Este alerta não requer que existam navios nessa área.

$$alarme.navio\{1,2\} = \begin{cases} 1 & selecionada\ a\ checkbox\ "Pesquisar\ lista\ de\ navios\ " \\ 0 & selecionada\ a\ checkbox\ "Pesquisa\ indiferenciada" \end{cases}$$

A opção "Pesquisar lista de navios" permite ao utilizador selecionar um navio ou vários através do seu MMSI. Esta opção é válida para criar alertas associados a contatos de interesse. Por exemplo, no alerta "madeira_tankers.mat" pretende-se alertar o utilizador quando navios de tipo "tanker" entrem na ZEE da Madeira entre as 10:00 e as 12:00, quaisquer que sejam os navios (desde que sejam do tipo tanker).

A componente (1,3) guarda a lista de navios introduzida referente à opção "Pesquisar lista de navios", enquanto que a componente (1,4) guarda o tipo de navio para pesquisa definido pelo utilizador na opção "Pesquisa indiferenciada". Cada uma das duas últimas componentes representa uma matriz 1x3 que guarda, respetivamente os dados referentes à velocidade (mínima e máxima) e rumo (mínimo e máximo) definidos pelo utilizador. O primeiro valor das matrizes 1x3, referentes à velocidade e rumo, representam uma variável binária que indica se o utilizador selecionou as respetivas *checkbox*[88] de velocidade e rumo, enquanto o segundo valor representa o valor mínimo introduzido e o terceiro representa o valor máximo introduzido.

- Field "metoc"

O campo "metoc" corresponde a um *cell array* de dimensão 1x2. À semelhança dos campos anteriores a componente (1,1) corresponde uma variável binária que representa se o alerta está associado a um quadro meteorológico ou não. A componente (1,2) representa um *cell array* de dimensão 4x4. A primeira coluna desta matriz indica a descrição das componentes seguintes. A coluna seguinte corresponde à variável binária utilizada para indicar se as *checkbox's* (TWS, TWD, SWH, WDir) foram selecionadas. Na terceira e quarta coluna estão representados, respetivamente, os valores mínimos e máximos definidos pelo utilizador.

```
>> alarme.metoc{1,2}

ans =

    'TWS'     [0]     [0]     [20]
    'TWD'     [0]     [0]     [20]
    'SWH'     [1]     [7]     [15]
    'WDir'    [0]     [0]     [20]
```

Figura 38 - Componente (1,2) do campo metoc (*cell array* 4x4).

[88] Função do MATLAB composta por uma variável binária que quando se encontra selecionada assume o valor de 1 e quando está desseleccionada adquire o valor 0.

- Field "área"

Este campo é constituído por um *cell array* de dimensão 1x4. A componente (1,1), representa a variável binaria que indica se o alerta tem integrado uma área de interesse. A componente (1,2) é composta por um *cell array* de dimensão 1x6. A primeira e segunda coluna desta matriz representam, respetivamente, o nome e sigla da área e as restantes colunas guardam as coordenadas da área. A componente (1,3) é utilizada para guardar o tipo de pesquisa na área, funcionando da seguinte forma:

$$alarme.area\{1,3\} = \begin{cases} 1 & selecionada\ a\ checkbox\ "Dentro\ da\ área" \\ 2 & selecionada\ a\ checkbox\ "Fora\ da\ área" \\ 3 & selecionada\ a\ checkbox\ "Fora\ da\ área\ a\ uma\ distância\ de" \end{cases}$$

Na componente (1,4) é guardado, caso seja selecionada a opção "Fora da área a uma distância de", o valor da distância.

- Field "*user*"

O campo "*user*" é utlizado para guardar o nome do utilizador que criou o alerta, no formato *string*.

- Field "*filename*"

Neste campo é guardado o nome do ficheiro do alerta, também no formato string.

- Field "data"

O campo "data" guarda o GDH em que o alerta foi criado no formato *datenum*.

- Field "dados"

Campo a ser usado futuramente para registar estatísticas associadas ao alerta.

- Field "*comments*"

Neste campo são guardados os comentários feitos pelo utilizador sobre o alerta, no formato *string*.

- Field "prioridade"

Este campo destina-se a guardar a prioridade do alerta. Após a seleção da prioridade pelo utilizador é guardado um valor (de 1 a 5) que representa a prioridade selecionada. As prioridades definidas encontram-se representadas no esquema a baixo indicado:

$$alarme.prioridade = \begin{cases} 1 & Muito\ alta \\ 2 & Alta \\ 3 & Média \\ 4 & Baixa \\ 5 & Muito\ baixa \end{cases}$$

3.3.2 Interface para construção de alertas

Como já foi anteriormente referido, este interface permite ao utilizador criar alertas, dispondo para isso de um conjunto de funcionalidades, associadas às dimensões do alerta, agrupadas em 5 telas distintas: Navio, GDH, Área, METOC e Dados do Alerta.

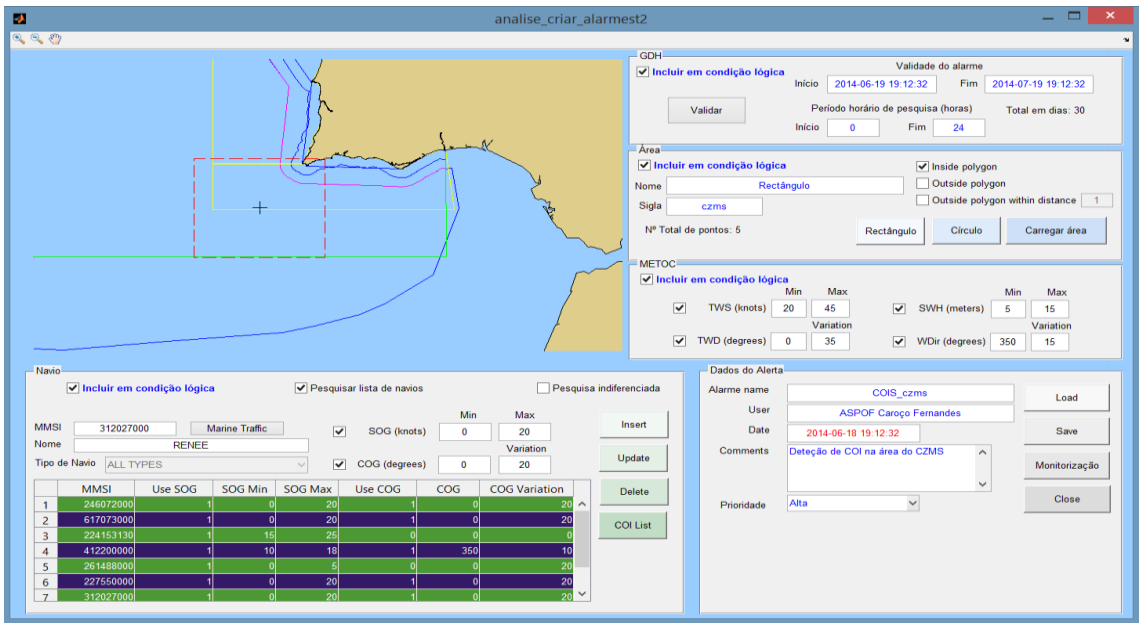


Figura 39 - Interface do módulo de criação de alertas.

No decorrer da implementação da estrutura de um alerta, optou-se por considerar que um alerta tem de incluir obrigatoriamente uma área. Esta opção tem impato nas rotinas que verificam se o evento de interesse está a ocorrer ou não. Desta forma, o primeiro filtro aos dados AIS seleciona as posições de navios que verificam a condição lógica selecionada pelo utilizador para a dimensão Área, o que reduz a quantidade de informação necessária para analisar nas rotinas associadas às restantes dimensões do alerta, traduzindo-se em ganhos de tempo computacional consideráveis.

Neste campo, o utilizador, pode carregar áreas já existentes ou criar uma nova área. Para esta última, o sistema dispõe de um mapa mundial, igual ao existente na aplicação AISINTEL, onde podem ser desenhadas áreas retangulares e circulares. Desta forma, "*são minimizados eventuais erros do operador na introdução das coordenadas (...), existindo também a vantagem do auxílio visual para melhor perceção da zona de análise*"[89]. Por outro lado podem ser carregadas áreas existentes, isto é, áreas geográficas pré-desenhadas com particular interesse para a Marinha. É ainda de salientar que o utilizador pode selecionar se pretende que o alerta se dê dentro ou nas proximidades da área.

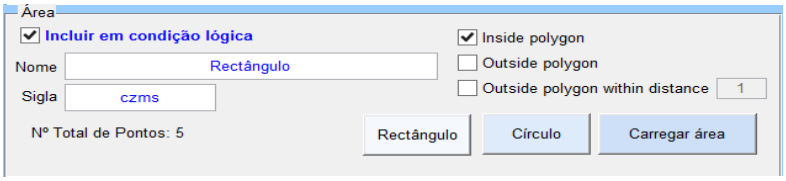


Figura 40 - Filtro da área.

Inicialmente é necessário definir a janela temporal, em que o alerta será válido. Esta é composta por dois parâmetros: a data de validade e período de validade. O primeiro corresponde à hora, dia, mês e ano em que o alerta é válido, pois este pode ser valido em apenas alguns períodos do ano. O segundo parâmetro diz respeito ao período do dia em que é valido, ou seja, o utilizador pode dentro da data de validade do alerta restringi-la a um período de algumas horas como mostra a imagem seguinte.

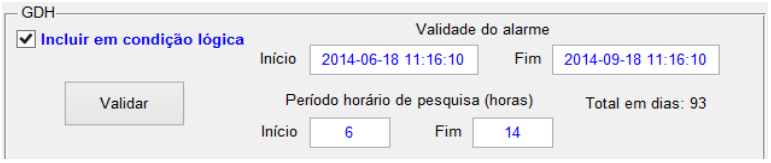


Figura 41 - Filtro temporal.

Outro parâmetro que pode ser incluído no alerta são as condições meteo-oceanográficas. Este campo possibilita, ao utilizador, a definição dos valores máximos e mínimos da velocidade do vento (TWS), direção do vento (TWD), altura significativa da onda (SWH) e sua direção (WDir).

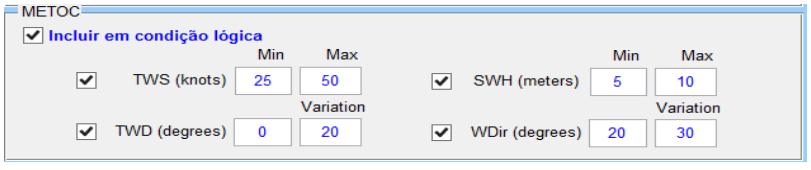


Figura 42 - Filtro das condições meteo-oceanográficas.

[89] MELO, Hugo Daniel Almeida de (2011). *Módulo de análise do quadro situacional marítimo para apoio a missões de vigilância e fiscalização marítima*, Dissertação de mestrado em Ciências Militares Navais - Marinha na Escola Naval: Marinha, pp.35.

No que diz respeito ao parâmetro lista de navios, como já foi referido, permite a criação de um filtro, de modo a detetar, na área selecionada, os navios ou tipo de navio que se pretende. Neste campo é possível pesquisar, na área, uma lista de navios ou efetuar uma pesquisa indiferenciada, isto é, pesquisar apenas pelo tipo de navio. Para realizar uma pesquisa por lista de navios, o utilizador, pode escolher listas COI, já pré-definidas, ou criar a sua própria lista através da inserção dos MMSI's dos respetivos navios. Esta aplicação permite ainda a criação de um filtro adicional para o rumo e velocidade dos navios da lista. Para executar uma pesquisa por tipo de navio, a aplicação, dispõe de um *pop-up menu*[90] com os seguintes tipos de navios: *tanker*, *cargo*, *passenger*, *law enforcement*, *fishing vessel*, *sailing vessel*, *other*, *tanker and cargo*, *tanker cargo and passeger* e *all types*. À semelhança da pesquisa por lista de navios, o utilizador, pode criar um filtro suplementar ao definir um rumo e velocidade mínimo e máximo.

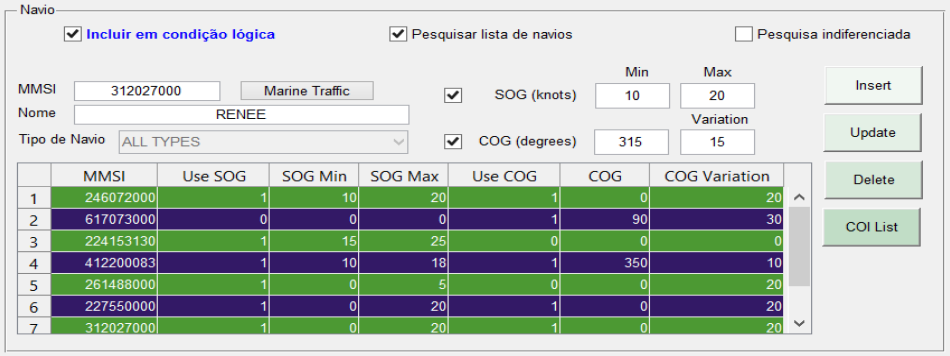


Figura 43 - Pesquisa por lista de navios.

Por último, no painel de dados do alerta, é possível atribuir um nome ao alerta, identificar o criador, fazer alguns comentários sobre o alerta e atribuir uma prioridade. Neste último campo é também possível guardar o alarme para posteriormente poder ser carregado na interface de monitorização.

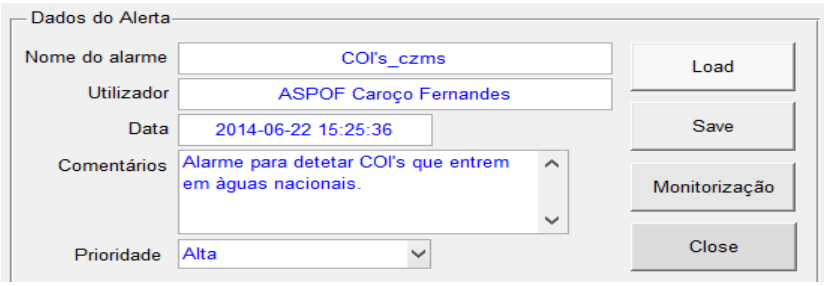


Figura 44 - Dados do alerta COI's_czms.

[90] O *pop-up menu* é uma ferramenta existente no MATLAB que disponibiliza ao utilizador uma lista de opções pré-definidas.

A prioridade é um atributo que está associado à cor do marcador (*marker*) usado para visualizar o alerta no mapa no interface de monitorização.

Tabela 3 - Prioridade associada a um alerta.

Prioridade	Cor
Muito alta	Vermelho
Alta	Laranja
Média	Amarelo
Baixa	Verde
Muito baixa	Branco

3.3.3 Interface para monitorização

Após a criação do alerta na interface de construção de alertas, torna-se necessário proceder à sua monitorização. Este interface permite efetuar a análise, no mesmo ciclo, de diferentes alarmes em simultâneo, bastando para isso carregar os alarmes desejados através do botão *"Load"*. É de realçar que neste interface é importante definir, antes de se proceder à monitorização dos alertas, a data de referência. Esta data poderá ser o GDH no presente momento ou um dia em especifico que o utilizador queira analisar. Esta funcionalidade permite ao utilizador replicar o alerta (verificar o comportamento do alerta numa janela de tempo do passado) caso seja necessário por motivos de validação do mesmo. Este interface integra um conjunto de rotinas, associadas às dimensões do alerta, para pesquisar se os atributos definidos são válidos ou não, e desta forma, alertar para a ocorrência do evento de interesse associado ao alerta.

Tabela 4 - Rotinas associadas a um alerta.

Rotina	Descrição	Estado
Fcn_pesquisar_alarme	Rotina que agrega outras rotinas para pesquisar a ocorrência de atributos relativos às dimensões de um alerta (fcn_alarme_pesquisar_gdh e fcn_alarme_pesquisar_navio, etc).	Concluído
Fcn_mostrar_alarmes_mapa	Esta rotina que permite mostra no mapa os contactos detetados pela interface de monitorização.	Concluído
Fcn_alarme_pesquisar_gdh	Rotina implementada com o intuito de filtrar as ocorrências de acordo com o período horário definido pelo utilizador.	Concluído
Fcn_alarme_pesquisar_navio	Esta rotina permite filtrar nas ocorrências pelo SOG e COG, assim como pelo tipo de navio.	Concluído
Fcn_alarme_pesquisar_metoc	Pesquisa na área definida, se as condições metoc estabelecidas pelo utilizador se verificam.	Incompleto

O ciclo de monitorização associado ao botão "monitorizar" está reproduzido no seguinte esquema:

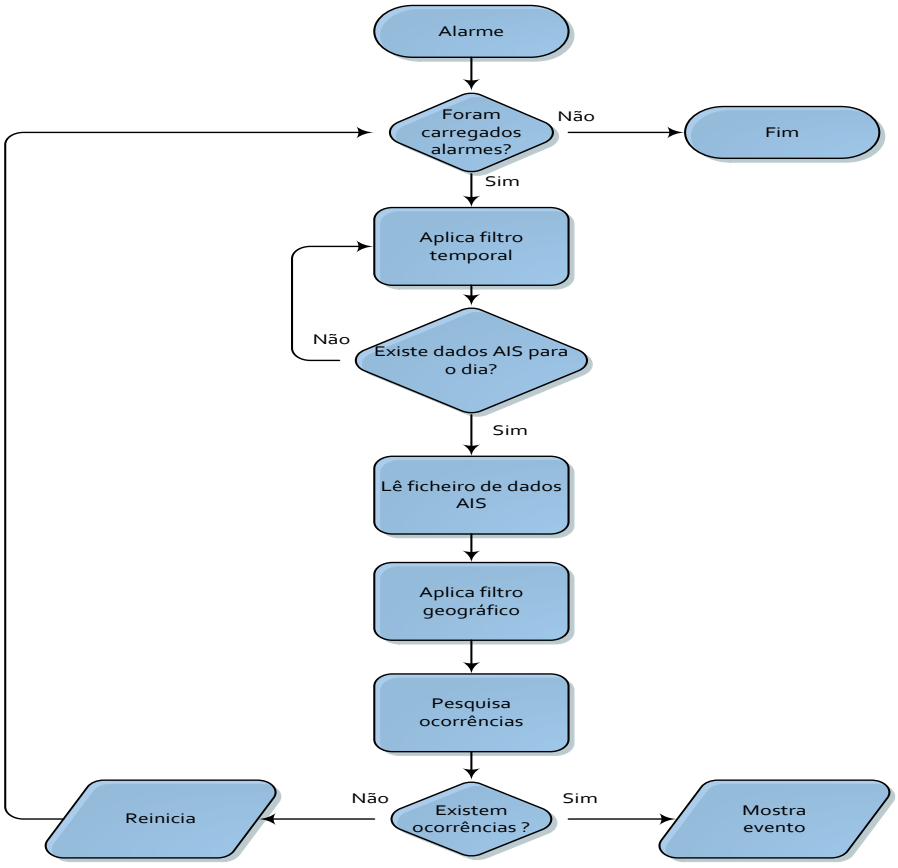


Figura 45 - Fluxograma da interface monitorizar.

Após ser pressionado o botão "Monitorizar", o ciclo iniciar-se-á podendo ser interrompido a qualquer momento pelo utilizador. Este pode também acompanhar na janela pesquisa a evolução do ciclo. Assim que houver alguma ocorrência aparecerá um contato na área definida no alerta, com a cor da prioridade atribuída ao alerta. Após o aparecimento de um contacto o utilizador pode ver informação (MMSI, posição, GDH da posição, velocidade, rumo) sobre o mesmo, para isso é apenas necessário carregar em cima do contacto com o botão direito do rato. Nas informações do contacto é disponibilizado o *link* para a base de dados do sistema *Marine Traffic* onde o utilizador terá informação mais detalhada do navio em causa. No campo das informações é também disponibilizado um *link* para o módulo Análise de Trajetória do AISINTEL.

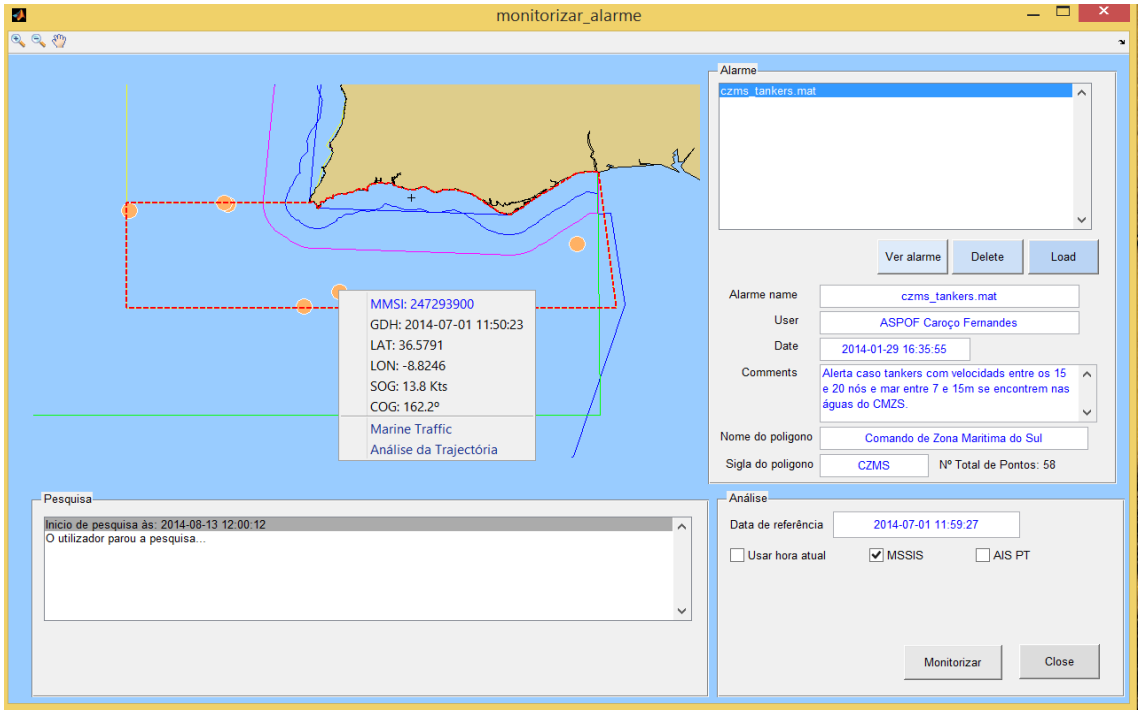


Figura 46 - Contatos detetados pelo alerta "czms_tankers.mat".

Este interface permite também, ao selecionar um alerta, regressar à interface de construção de alertas de modo a efetuar as alterações que o utilizador achar conveniente.

3.4 Métricas para avaliação de alertas

Os alertas criados podem incidir num evento em que o operador/utilizador apenas esteja interessado em ser avisado desse fato uma única vez, que coincide com a sua primeira ocorrência. Por exemplo, considere-se a situação em que o operador do módulo de alertas pretende ser avisado do momento em que um determinado navio (previamente

identificado para o efeito) com carga perigosa entre em águas territoriais, para de seguida proceder ao seu seguimento. O alerta criado para este efeito tem um propósito que se esgota no momento em que o alerta é dado. O importante nesta situação está na rapidez com que o sistema efetua os cálculos necessários para validar os atributos associados ao evento de interesse e na sua comunicação ao utilizador.

Outro tipo de situação prende-se com eventos que poderão ter interesse em serem detetados ao longo do tempo, sempre que ocorram. Este tipo de eventos estão relacionados, em geral, com situações de risco de segurança marítima. Por exemplo, o operador/analista poderá estar interessado, não só, em ser avisado de navios que se aproximem de áreas consideradas perigosas ou de alto risco para a navegação (zonas costeiras referenciadas de alto risco pela sua topografia), mas também, conhecer a frequência com que este tipo de evento ocorre ao longo de um ano ou qualquer outro período de análise.

Nas duas situações referidas atrás interessa caracterizar, por um lado, a eficiência computacional associada às rotinas que avaliam os atributos do alerta, e por outro lado, a frequência com que determinada situação de risco ocorre. Em ambas as situações serão usadas métricas para atingir os propósitos definidos. As métricas podem ser divididas em métricas diretas e indiretas. As primeiras dizem respeito a uma medida realizada em termos de atributos observados, enquanto que as métricas indiretas referem-se a medidas obtidas a partir de outras métricas (Vasconcelos A., 2005). Todas as métricas apresentadas neste trabalho são métricas diretas.

Figura 47 - Perspetivas/planos de avaliação de métricas.

3.4.1 Métricas para avaliação da eficiência computacional

Para avaliar a eficiência computacional dos alertas foram definidas as seguintes métricas:

- E_1 – Tempo decorrido na execução de uma rotina em segundos;
- E_2 – Tempo decorrido num ciclo da pesquisa do alerta.

A primeira métrica pode ser obtida recorrendo às funções etime, tic, toc ou cputime do MATLAB. O objetivo consiste em medir o tempo que uma rotina demora a ser executada durante um ciclo de pesquisa de um alerta. As rotinas estão identificadas na tabela 4. A segunda métrica pretende medir o tempo total de um ciclo de pesquisa de um alerta, que não corresponde necessariamente à soma dos tempos de execução das rotinas que são executadas dentro desse ciclo.

Estas métricas são úteis para identificar instruções ou partes do código associado às rotinas que poderão ser objeto de melhoramento. A partir destas métricas diretas poderão ser definidas métricas indiretas que poderão medir a eficiência das rotinas implementadas.

3.4.2 Métricas para caraterizar ocorrência de eventos de interesse

De modo a caracterizar as ocorrências dos eventos de interesse associados a um alerta, foram identificadas as seguintes métricas:

- E'_1 – Num período de k dias (janela de tempo entre o dia inicial e final do alerta) o número de ocorrências de um alerta;
- E'_2 – Num período de k dias o número de contactos identificados pelo alerta;
- E'_3 – Num período de k dias a velocidade média dos contatos identificados pelo alerta;
- E'_4 – Num período de k dias o rumo dominante dos contatos;
- E'_5 – Num período de k dias a moda do dia da semana dos contatos;
- E'_6 – Num período de k dias a moda do período sinóptico dos contatos;
- E'_7 – Num período de k dias a média das condições meteo-oceanográficas, medidas na escala de Beaufort, no local dos contatos identificados pelo alerta.

Estas métricas permitem caracterizar a frequência com que determinados eventos de interesse ocorrem e relacionar estes eventos com outras peças de informação (por exemplo intensidade de tráfego marítimo) de forma a realizar estudos mais detalhados no âmbito da segurança marítima.

A partir destas métricas poderão ser obtidas métricas indiretas como, por exemplo, um índice de risco associado a uma área para a qual foi definido um alerta, tendo este sido monitorizado e objeto de medição e registo durante um determinado período de tempo.

Capítulo 4

Discussão de Resultados

4.1 Monitorização e avaliação de alertas

4.2 Análise de resultados e comparação das métricas

4 Capítulo 4 – Discussão de Resultados

Neste capítulo pretende-se testar um conjunto de alertas criados na sequência dos requisitos operacionais especificados pelo CADOP. Por conseguinte definiu-se um conjunto de três alertas referentes a eventos de interesse distintos relacionados com segurança marítima.

4.1 Monitorização e avaliação de alertas

Com o objetivo de testar as rotinas implementadas para efetuar a monitorização foram criados três alertas que poderão constituir situações de interesse para a comunidade operacional. A tabela 5 resume os três alertas criados:

Tabela 5 - Alertas testados.

Alerta	Descrição	Tipo de navio	Área
EST_SV	Deteção de navios a transitar na zona de separação de tráfego[91] de São Vicente com velocidade inferior a 30 nós.	Todos os tipos	Esquema de separação de tráfego de São Vicente
CZMS_tankers3	Deteção de *tankers* que entrem no CZMS com velocidade inferior a 10 nós, entre as 20:00 e 24:00.	*Tankers*	CZMS
MT_pesca	Deteção de navios de pesca a operar no mar territorial com velocidade inferior a 5 nós, entre as 20:00 e 24:00	*Fishing vessels*	Mar territorial[92]

[91] O primeiro alerta destina-se a verificar a existência de navios nas zonas de separação de tráfego, que segundo o Regulamento Internacional para Evitar Abalroamentos no Mar (RIEAM) um navio "*não deve penetrar na zona de separação ou cruzar a linha de separação, exceto: (i) em caso de emergência, para evitar um perigo imediato; (ii) para pescar na zona de separação*" (de acordo com a alínea e) da regra nº10 do Regulamento Internacional para Evitar Abalroamentos no Mar).

[92] O mar territorial é uma zona de mar adjacente, medida desde a linha base até a um limite de máximo de 12 milhas naúticas.

Os alertas acima referidos foram monitorizados entre 1 e 10 de julho de 2014 num PC portátil com um processador Intel® Core™ i7-36730QM CPU @ 2.40GHz com 8.00GB de memória RAM. Para o efeito, foram cedidos pela DAGI, no âmbito desta dissertação, os dados AIS referentes ao período de 1 a 10 de julho de 2014.

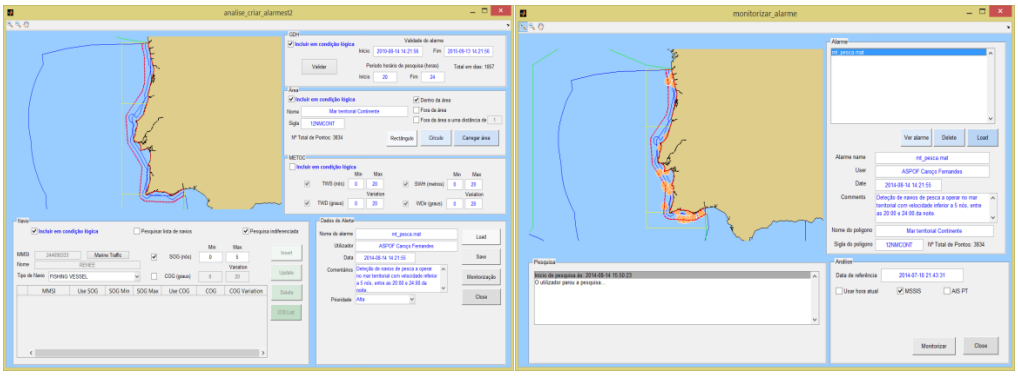


Figura 48 – Criação e monitorização do alerta MT_PESCAS.

A tabela seguinte apresenta o número de contactos identificados pelo sistema de monitorização em cada dia do período de análise e respetiva velocidade média:

Tabela 6 – Número de contatos e velocidade média (em nós) no período de 1 a 10 de julho de 2014.

Alerta\Dia	1	2	3	4	5	6	7	8	9	10
EST_SV	5 (5,1)	5 (4,2)	4 (10,8)	6 (3,7)	5 (3,2)	5 (5,2)	5 (3,4)	3 (2,5)	1 (3,4)	3 (8,3)
CZMS_tankers3	1 (8,5)	1 (9,8)	0	2 (9,7)	2 (9,3)	0	0	4 (7,7)	2 (8,0)	2 (8,6)
MT_pesca	24 (1,6)	19 (2,2)	14 (1,8)	16 (2,4)	15 (2,9)	13 (2,7)	23 (2,1)	23 (2,3)	26 (2,4)	27 (1,7)

A tabela seguinte apresenta o tempo decorrida num ciclo de monitorização para cada um dos alertas no período de análise:

Tabela 7 – Tempo (em segundos) de um ciclo de monitorização no período de 1 a 10 de julho de 2014.

Alerta\Dia	1	2	3	4	5	6	7	8	9	10
EST_SV	0,0150	0,0220	0,0090	0,0190	0,0230	0,0210	0,0340	0,0150	0,0460	0,0070
CZMS_tankers3	3,3720	3,0860	2,3980	2,9850	2,9840	2,0490	2,6500	2,7480	3,7440	2,9730
MT_pesca	5,5100	5,1200	4,9480	5,0550	4,8700	4,4150	4,6590	4,7480	5,7110	5,9740

O primeiro alerta destingui-se dos restantes por não colocar nenhuma especificação (restrição) no tipo de navio. Isto significa que nos alertas CZMS_tankers3 e MT_pesca é necessário averiguar o tipo de navio que se encontra no interior do polígono da área respetiva. Também o primeiro alerta não coloca nenhuma restrição ao período de pesquisa (os restantes alertam definem a janela de tempo entre as 20:00 e as 24:00).

4.2 Análise de resultados e comparação de métricas

Observando as tabelas da secção anterior verifica-se que os alertas CZMS_tankers3 e MT_pescas demoraram mais tempo em média que o alerta EST_SV. Os dois últimos alertas diferem do primeiro pelo facto de especificarem o tipo de navio. Esta especificação implica que durante a monitorização seja evocada um conjunto de instruções na rotina fcn_alarme_pesquisar_navio para verificar e seleccionar navios do tipo definido no alerta. Esta execução extra de instruções consome, naturalmente, mais tempo, do que na situação em que não se coloca esta restrição de tipo. Entende-se que este é o motivo pelo qual os dois últimos alertas demoram mais tempo que o primeiro.

Observando os tempos entre o último e o penúltimo alertas verifica-se que o último demora, em média, mais 2 segundos. Neste caso, entende-se que o número de ocorrências poderá explicar o tempo extra. O gráfico seguinte apresenta o cruzamento destes valores para ambos os alertas:

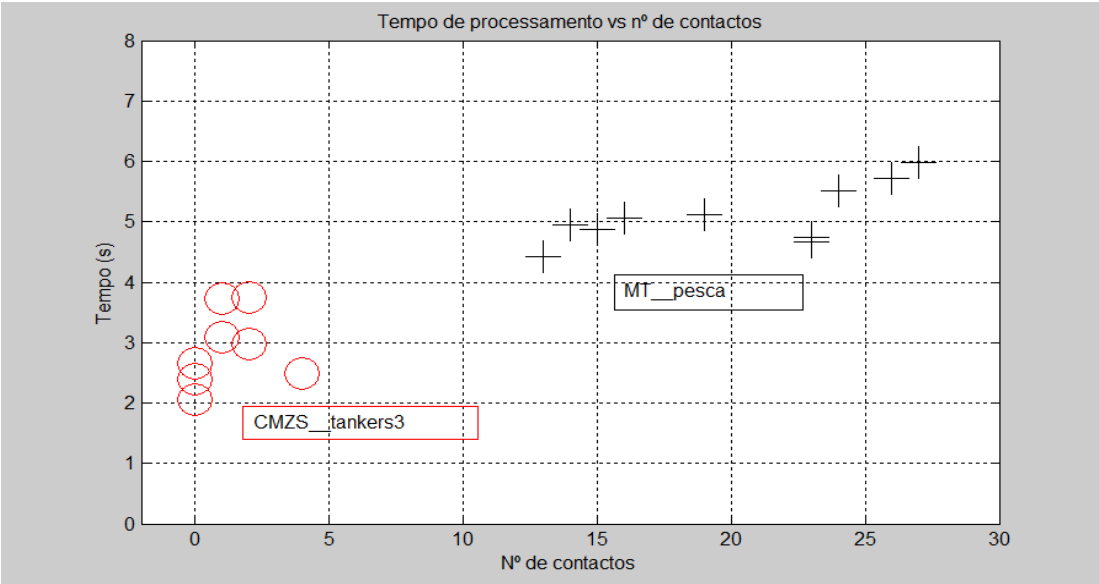


Gráfico 1 – Tempo de processamento vs número de contatos detetados.

O número de contatos detetados pelo alerta constitui um listagem única de navios distintos que satisfazem as condições lógicas associadas aos atributos do respetivo alerta. Sucede que durante a pesquisa de contactos é feita uma listagem de todas as posições de cada um dos contactos. Sobre esta listagem obtém-se, para cada contacto, a posição mais recente. Entende-se que manipulação em memória da primeira listagem consome mais tempo quanto maior for a sua dimensão. Naturalmente, a sua dimensão (nº de mensagens AIS) depende do número de contatos detetados. Esta diferença de dimensões nesta primeira listagem explica a diferença de cerca de 2 segundos entre os tempos de processamento do penúltimo para o último alerta.

Na opinião do autor, será interessante efetuar uma análise mais completa, recorrendo às métricas diretas definidas na secção 3.4, no sentido de efetuar um estudo mais rigoroso das rotinas e do efeito dos atributos do alerta nos tempos de processamento destes. Face ao tempo disponível, não foi possível efetuar esta análise na sua globalidade e totalidade.

Capítulo 5

Conclusões e Recomendações

5.1 Análise sumária do trabalho realizado

5.2 Recomendações e trabalho futuro

5 Capítulo 5 – Conclusões e Recomendações

Neste capítulo pretende-se realizar uma análise sumária do trabalho realizado, assim como dos resultados obtidos. Importa igualmente, nesta fase, refletir sobre os objetivos alcançados, questões de investigação colocadas e trabalhos futuros que possam resultar do presente trabalho.

5.1 Análise sumária do trabalho realizado

No que diz respeito aos objetivos definidos para esta dissertação, de acordo com o apresentado nos capítulos 3 e 4, entende-se que foram na sua maioria alcançados de forma bastante satisfatória. Apenas o terceiro objetivo não foi totalmente atingido, pois apenas foi analisado uma métrica de avaliação da eficiência computacional (tempo decorrido num ciclo de monitorização) e duas métricas de avaliação dos alertas (velocidade média e nº de contatos identificados pelo alerta). Face à falta de tempo não foi possível realizar a análise das restantes métricas, ficando como sugestão para trabalhos futuros.

Relativamente às questões de investigação levantadas, considera-se que as questões expressas no primeiro capítulo foram na sua maioria alvo de estudo e desenvolvimento nesta dissertação. As duas primeiras questões foram respondidas nos capítulos 3 e 4, ficando por responder à terceira questão. Isto deveu-se ao fato de esta questão requerer um período de tempo considerável, para analisar e caraterizar as SRR's nacionais.

Em suma, a presente dissertação permite afirmar que:

- Foram identificados e propostos um conjunto de atributos que caraterizam um alerta;

- Desenvolvidas rotinas para construção e monitorização de alertas, que permitam a deteção de eventos de interesse;

- Foram definidas métricas que permitem avaliar o módulo de alertas ao nível da eficiência computacional e caraterizar as ocorrências de eventos de interesse.

- Verificou-se um aumento do tempo de processamento quando se aumenta as especificações do alerta. Para não prejudicar a performance da monitorização, as rotinas associadas ao teste lógico dos atributos especificados deverão ser objeto de otimização e prioridade em termos de desenvolvimento.

É de realçar que todo o empenho para a realização desta dissertação seria inútil sem a ajuda de duas entidades como a DAGI e CADOP. A cooperação estabelecida entre o orientador, como crítico operacional do estudo realizado, e o coorientador, como técnico e indispensável auxílio "*em todas as questões de programação e análise levantadas*"[93], permitiram alcançar os objetivos propostos. Desta forma espera-se que este estudo seja prosseguido pela DAGI através do incentivo à utilização do mesmo pela comunidade operacional.

5.2 Recomendações e trabalho futuro

O presente estudo reflete o empenho que a Marinha tem vindo a realizar, no sentido de inovar e procurar constantemente melhorar a sua capacidade de CSM.

A nível operacional, o módulo desenvolvido na presente dissertação revela-se uma inovação, constituindo uma mais-valia para a comunidade operacional. Não obstante, esta aplicação necessita de ser validada num ambiente operacional. Para o efeito, o COMAR revela-se no local ideal para a validação desta ferramenta, possuindo todas as condições para identificação de possíveis lacunas e deficiências da aplicação. Deste modo, no que diz respeito ao nível operacional, recomenda-se que sejam tomadas as seguintes ações:

- Efetuar a validação do módulo de alertas no COMAR;

- Implementação do módulo de alertas na ferramenta *Oversee* e AISINTEL.

O presente estudo, assim como as dissertações precedentes (Melo L.,2010; Melo H.,2011; Sousa L.,2013), abre portas para a exploração de novas funcionalidades e capacidades na área da alarmística, possibilitando assim usufruir de todo o potencial dos dados AIS. No entanto é possível desenvolver muitos outros estudos, que tenham

[93] MELO, Hugo Daniel Almeida de (2011). *Módulo de análise do quadro situacional marítimo para apoio a missões de vigilância e fiscalização marítima*, Dissertação de mestrado em Ciências Militares Navais - Marinha na Escola Naval: Marinha, pp.88.

utilidade operacional. Assim, no que ao plano académico diz respeito, sugere-se as seguintes linhas de ação:

- Implementação de rotinas para dados meteo-oceanográficos;

- Desenvolvimento de uma estrutura de dados que permite o registo de estatísticas relativas à monitorização do alerta;

- Realização da avaliação dos alertas com maior profundidade usando para o efeito um conjunto mais alargado de métricas diretas, quer no plano da eficiência computacional, como no plano da caracterização do evento de interesse associado ao alerta;

- Caraterização da área marítima de responsabilidade nacional, através do desenvolvimento de rotinas que permitam quantificar e caracterizar os navios que navegam nessas águas.

- Criação de métricas indiretas na área do CSM como por exemplo índices de risco por tipo de evento de interesse.

Referências Bibliográficas

Referências Bibliográficas

6 Bibliografia

BRITO, B. M. (2011). *Cálculo de áreas de busca oceânicas com incerteza na posição e hora de ocorrência.* Dissertação de mestrado em Ciências Militares Navais - Ramo Marinha na Escola Naval: Marinha.

CRITICAL SOFTWARE. (2014). *Intelligent Operations Centre*. Obtido em 31 de março de 2014, de http://www.criticalsoftware.com/uploads/resources/20140331%20-%20Oversee%20-%20Flyer%20A4_20140403155214.pdf?v34

CRITICAL SOFTWARE. (s.d.). *Oversee*. Obtido em 10 de outubro de 2013, de http://www.criticalsoftware.com/rd/

DECRETO-LEI nº233/2009 de 15 de setembro. *Diário da República nº179 - 1ª série.* Ministério da Defesa Nacional.

DESPACHO do ALM CEMA nº18/09 de 18 de maio. *Ordem da Armada.* Marinha.

EMGFA. (s.d.). *NATO - Ocean Shield*. Obtido em 26 de julho de 2014, de http://www.emgfa.pt/pt/operacoes/missoes/oceanshield

EMGFA. (s.d.). *NATO OAE (Operation Active Endeavour)*. Obtido em 26 de julho de 2014, de http://www.emgfa.pt/pt/operacoes/missoes/oae-mediterraneo

EMSA. (2009). *An information system to improve maritime safety in Europe.* [s.l.]: EMSA.

EMSA. (2014). *Integrated Maritime Data Environment*. Obtido em 6 de fevereiro de 2014, de http://emsa.europa.eu/operations/maritime-monitoring/86-maritime-monitoring/1520-integrated-maritime-data-environment-imdate.html

GAMEIRO MARQUES, C.-a. (Março de 2013). Cibersegurança e conhecimento situacional marítimo. *in Revista da Armada*, p. 13.

LOPES, J. C. (2013). *Discursos do Chefe do Estado-Maior da Armada*. Obtido em 5 de fevereiro de 2014, de http://www.marinha.pt/pt-pt/media-center/noticias-destaques/Paginas/Discurso-Chefe-Estado-Maior-Armada.aspx

MACEDO, F. W., & SARDINHA, A. M. (1987). *Fogos Florestais* (Vol. 2º). Lisboa: Publicações Ciência e Vida.

MARINE TRAFFIC. (2014). *Marine Traffic*. Obtido em 2 de maio de 2014, de https://www.marinetraffic.com/pt/ais/home/?lang=pt

MARINHA. (2007). *Regulamento internacional para evitar albaroamentos no mar - 1972* (7 ed.). Lisboa: Marinha.

MARINHA. (2011). *Directiva Sectorial da Superintendência dos Serviços de Tecnologias da Informação 2011.* Lisboa: Marinha.

MARINHA. (2011). *Diretiva de Política Naval 2011.* Lisboa: Marinha.

MARINHA. (2012). *IONAV 1010 - Relatos e comunicados operacionais.* Lisboa: Marinha.

MARINHA. (2014). *Diretiva de Planeamento Naval 2014.* Lisboa: Marinha.

MARINHA. (s.d.). *Portugal uma nação marítima*. Obtido em 3 de outubro de 2013, de http://www.marinha.pt/pt-pt/historia-estrategia/estrategia/folhetospt/Portugal_uma_nacao_maritima.pdf

MELO, H. D. (2011). *Módulo de análise do quadro situacional marítimo para apoio a missões de vigilância e fiscalização marítima.* Dissertação de mestrado em Ciências Militares Navais - Ramo Marinha na Escola Naval: Marinha.

MELO, L. C. (2010). *Indicadores de conhecimento situacional do espaço de envolvimento marítimo com recurso aos dados AIS.* Dissertação de mestrado em Ciências Militares Navais - Ramo Marinha na Escola Naval: Marinha.

NATO. (s.d.). Obtido em 26 de julho de 2014, de http://www.arrc.nato.int/Unified_Protector.aspx

NATO. (2008). *NATO Glossary (AAP6).* [s.l.]: NATO.

NATO. (2009). *MCCIS 6.0.0 system user manual.* [s.l.]: NATO.

NATO. (2011). *Project 0IS0308100 - Provide functional services for command control of maritime operations.* Norfolk: NATO.

Oates, B. J. (2006). *Researching Information Systems and Computing.* London: SAGE Publications Ltd.

PORTO EDITORA. (2014). *Infopédia*. Obtido em 22 de março de 2014, de http://www.infopedia.pt/lingua-portuguesa/alerta

PROTEÇÃO CIVIL. (s.d.). *Normas para conceção do sistema de alerta e aviso no âmbito dos planos de emergência interna de barragens.* Obtido em 22 de março de 2014, de http://www.proteccaocivil.pt/RiscosVulnerabilidades/RiscosNaturais/SegurancaBarragens/Documents/NORMAS%20Barragens.pdf

SOMMERVILLE, I. (2011). *Software Engineering* (9th ed.). Boston: Addison-Wesley.

SOUSA, L. S. (2013). *Indicadores de risco de incidentes marítimos com base em dados do sistema de monitorização contínua das atividades de pesca.* Dissertação de mestrado em Ciências Militares Navais - Ramo Marinha na Escola Naval: Marinha.

UNIVERSIDADE EGEU. (s.d.). *Dimitrios Lekkas*. Obtido em 12 de agosto de 2014, de http://www.syros.aegean.gr/users/lekkas/

VASCONCELOS, A. (2005). *Metricas de Software*. Obtido em 1 de Agosto de 2014, de http://www.cin.ufpe.br/~if720/slides/introducao-a-metricas-de-software.ppt

VOLPE CENTER. (s.d.). *MSSIS Maritime Safety and Security Information System*. Obtido em 18 de outubro de 2013, de https://mssis.volpe.dot.gov/Main/overview/

VOLPE CENTRE. (s.d.). *Enabling Global Maritime Situational Awareness*. Obtido em 10 de outubro de 2013, de https://www.google.pt/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CC8QFjAA&url=http%3A%2F%2Fhandle.dtic.mil%2F100.2%2FADA519835&ei=k0llUvSVE5Kihgflj4CYAg&usg=AFQjCNGEEvkwpduMJB7de8Fn2VW6NViR-Q&sig2=YWYdx-v6C8gvfsq8rgRJ1A&bvm=bv.54934254,d.ZG4

Anexos

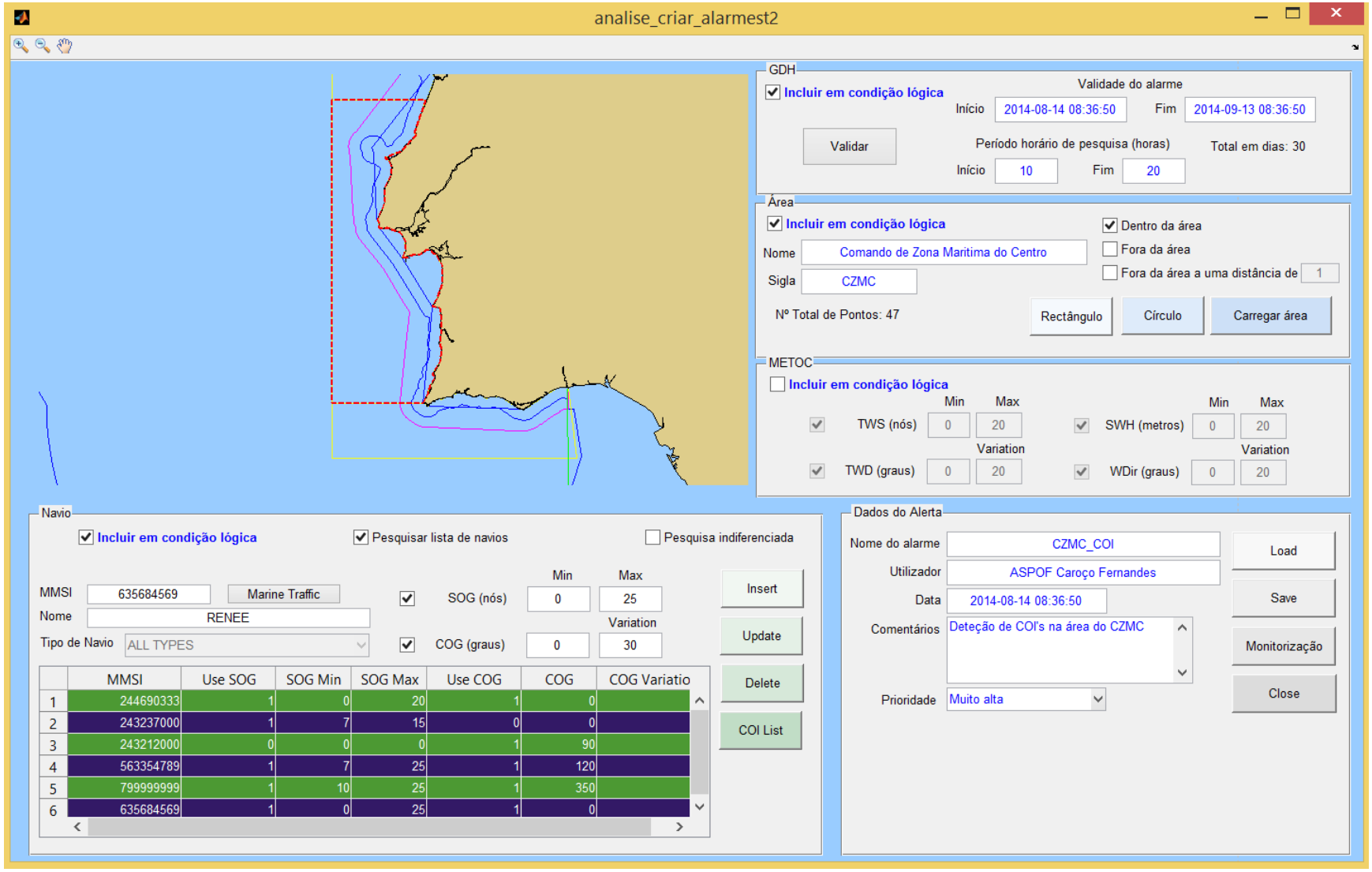
A – Interface de construção de alertas

B – Script da interface de construção de alertas

C – Interface de monitorização

D – Script da interface de monitorização

Anexo A – Interface de construção de alertas



Anexo B – Script da interface de construção de alertas

```
function varargout =
analise_criar_alarmest2(varargin)
% ANALISE_CRIAR_ALARMEST2 MATLAB code
for analise_criar_alarmest2.fig
%      ANALISE_CRIAR_ALARMEST2, by
itself, creates a new
ANALISE_CRIAR_ALARMEST2 or raises the
existing
%      singleton*.
%      H = ANALISE_CRIAR_ALARMEST2
returns the handle to a new
ANALISE_CRIAR_ALARMEST2 or the handle
to
%      the existing singleton*.
%
ANALISE_CRIAR_ALARMEST2('CALLBACK',hOb
ject,eventData,handles,...) calls the
local
%      function named CALLBACK in
ANALISE_CRIAR_ALARMEST2.M with the
given input arguments.
ANALISE_CRIAR_ALARMEST2('Property','Va
lue',...) creates a new
ANALISE_CRIAR_ALARMEST2 or raises the
%      existing singleton*.  Starting
from the left, property value pairs
are
%      applied to the GUI before
analise_criar_alarmest2_OpeningFcn
gets called.  An
%      unrecognized property name or
invalid value makes property
application
%      stop.  All inputs are passed to
analise_criar_alarmest2_OpeningFcn via
varargin.
%      *See GUI Options on GUIDE's
Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the
response to help
analise_criar_alarmest2
% Last Modified by GUIDE v2.5 29-Jan-
2014 15:04:07
% Begin initialization code - DO NOT
EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',
mfilename, ...
'gui_Singleton',  gui_Singleton,
'gui_OpeningFcn',
@analise_criar_alarmest2_OpeningFcn,
...
'gui_OutputFcn',
@analise_criar_alarmest2_OutputFcn,
'gui_LayoutFcn',  [] , ...
  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback =
str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] =
gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State,
varargin{:});
end
% End initialization code - DO NOT
EDIT

% --- Executes just before
analise_criar_alarmest2 is made
visible.
function
analise_criar_alarmest2_OpeningFcn(hOb
ject, eventdata, handles, varargin)
% This function has no output args,
see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% varargin   command line arguments to
analise_criar_alarmest2 (see VARARGIN)
% Choose default command line output
for analise_criar_alarmest2
handles.output = hObject;
%%%%MAPAS %%%%%%%%
%handles.axes1_latlim = [16.5 50];
%handles.axes1_lonlim = [-55 10];
%handles.axes1_latlimp = [16.5 50];
%handles.axes1_lonlimp = [-55 10];
handles.axes1_latlim = [-16.5 60];
handles.axes1_lonlim = [-75 50];
handles.axes1_latlimp = [-16.5 60];
handles.axes1_lonlimp = [-75 50];
%directoria para a pasta de dados
if
exist('caminho_dados_ais.mat','file')
    load caminho_dados_ais%carrega
dir_dados
setappdata(handles.pushbutton1,'dir_da
dos',dir_dados);%'C:\Documents and
Settings\user\Os meus
documentos\MATLAB\trabalho
else
    dir_dados =
uigetdir(pwd,'Directoria para a pasta
de dados (BDAIS)');
setappdata(handles.pushbutton1,'dir_da
dos',dir_dados);
    save caminho_dados_ais dir_dados
```

```
end
%directoria para a pasta MONICAP
if
exist('caminho_dados_monicap.mat','fil
e')
    load caminho_dados_monicap%carrega
dir_trabalho
setappdata(handles.pushbutton1,'dir_mo
nicap',dir_monicap);%'C:\Documents and
Settings\user\Os meus
documentos\MATLAB\trabalho
else
    dir_monicap =
uigetdir(pwd,'Directoria para a pasta
de dados MONICAP');

setappdata(handles.pushbutton1,'dir_mo
nicap',dir_monicap);
    save caminho_dados_monicap
dir_monicap
end
%directoria para a pasta FOTOS
if exist('caminho_fotos.mat','file')
    load caminho_fotos%carrega
dir_fotos
setappdata(handles.pushbutton1,'dir_fo
tos',dir_fotos);%'C:\Documents and
Settings\user\Os meus
documentos\MATLAB\trabalho
else
    dir_fotos =
uigetdir(pwd,'Directoria para a pasta
de FOTOS');
setappdata(handles.pushbutton1,'dir_fo
tos',dir_fotos);
    save caminho_fotos dir_fotos
end
%directoria para a pasta de trabalho
if
exist('caminho_trabalho_ais.mat','file
')
    load caminho_trabalho_ais%carrega
dir_trabalho
setappdata(handles.pushbutton1,'dir_tr
abalho',dir_trabalho);%'C:\Documents
and Settings\user\Os meus
documentos\MATLAB\trabalho
else
    dir_trabalho =
uigetdir(pwd,'Directoria para a pasta
de trabalho');

setappdata(handles.pushbutton1,'dir_tr
abalho',dir_trabalho);
    save caminho_trabalho_ais
dir_trabalho
end
%axes(handles.axes1)
set(gcf, 'Renderer', 'painters');
%ax=worldmap(handles.axes1_latlim,hand
les.axes1_lonlim);set(ax,'tag','mapa')
;
ax=worldmap('world');
setm(ax,'mapprojection','eqdcylin')
%setm(ax,'FFaceColor',([0.941 0.941
0.941]),'Grid','on')
setm(ax,'FFaceColor',([0.6 0.8
1]),'Grid','on')
load ([dir_trabalho
'\poligonos_reduced_c200']);%
warning off
h=patchesm1(latreduced2,lonreduced2,10
,[222/255 205/255 139/255]);
%load ('C:\Documents and
Settings\user\Os meus
documentos\MATLAB\trabalho\poligono_ma
pa');
%h=patchesm(poligono_mapa(:,1),poligon
o_mapa(:,2),10,[222/255 205/255
139/255]);
%load ('C:\Documents and
Settings\user\Os meus
documentos\MATLAB\trabalho\poligono_ma
pas_reduced');
%h=patchesm(lat,lon,10,[222/255
205/255 139/255]);
set(h,'Tag','coast');
load([dir_trabalho
'\tabela_areas.mat'])
load([dir_trabalho '\tabela_mes.mat'])
h2=plotm(tabela_areas{2,3},tabela_area
s{2,4},-2);
set(h2,'color','k');
h3=plotm(tabela_areas{1,3},tabela_area
s{1,4},-2);
set(h3,'color','b');
h4=plotm(tabela_areas{3,3},tabela_area
s{3,4},-2);
set(h4,'color','m');
h5=plotm(tabela_areas{6,3},tabela_area
s{6,4},-2);
set(h5,'color','y');
h6=plotm(tabela_areas{7,3},tabela_area
s{7,4},-2);
set(h6,'color','y');
h7=plotm(tabela_areas{8,3},tabela_area
s{8,4},-2);
set(h7,'color','y');
h8=plotm(tabela_areas{9,3},tabela_area
s{9,4},-2);
set(h8,'color','g');
h9=plotm(tabela_areas{10,3},tabela_are
as{10,4},-2);
set(h9,'color','g');
%h10=linem([45;-40],[29;-40]);
%set(h10,'color','g');
setm(ax,'fontsize',6)
tightmap
%%%%%%%%%  FIM MAPAS
%Preencher lista de peioridade do
alarme
str{1,1}='Muito alta';str{2,1}='Alta';
str{3,1}='Média';str{4,1}='Baixa';
str{5,1}='Muito baixa';
set(handles.popupmenu2,'string',str,'v
alue',3,'foregroundcolor','r')
prioridade=[];
setappdata(handles.pushbutton1,'priori
dade',prioridade)
varargin
if isempty(varargin)
%%%% GDH diferença de tempo %%%%
    d=now;
set(handles.edit26,'string',datestr(d,
31),'Foregroundcolor','r')
```

```
set(handles.edit27,'string',datestr(d+
30,31),'Foregroundcolor','r')
set(handles.edit28,'string',num2str(0)
,'Foregroundcolor','r')
set(handles.edit29,'string',num2str(24
),'Foregroundcolor','r')
set(handles.text33,'string','Total em
dias: 30')
set(handles.edit1,'string',num2str(244
690333))
set(handles.edit2,'string','RENEE')
%%%%% Estrutura dados NAVIO %%%%%
%Preencher lista com tipos de navios--
---------
    str{1,1}='ALL
TYPES';str{2,1}='TANKER';
str{3,1}='CARGO';str{4,1}='PASSENGER';
    str{5,1}='LAW
ENFORCEMENT';str{6,1}='FISHING
VESSEL';
    str{7,1}='SAILING
VESSEL';str{8,1}='OTHER';
    str{9,1}='TANKER AND
CARGO';str{10,1}='TANKER, CARGO AND
PASSENGER';
set(handles.popupmenu1,'string',str)
set(handles.popupmenu1,'enable','off')
    lista_navios=[];
setappdata(handles.pushbutton1,'lista_
navios',lista_navios)
lista_navios_indiferenciada=[];
setappdata(handles.pushbutton1,'lista_
navios_indiferenciada',lista_navios_in
diferenciada)
set(handles.edit23,'string','Inserir_n
ome_poligono','Foregroundcolor','r')
set(handles.edit25,'string','Inserir_S
igla','Foregroundcolor','r')
set(handles.edit7,'string',['Inserir_n
ome_do_alarme' datestr(d,30)
'.mat'],'Foregroundcolor','r')
set(handles.edit8,'string',datestr(d,3
1),'Foregroundcolor','r')
set(handles.edit9,'string','Inserir_po
sto_e_nome','Foregroundcolor','r')
set(handles.edit10,'string','Inserir_c
omentário','Foregroundcolor','r')
navio{1,1}=get(handles.checkbox1,'valu
e');
    if
get(handles.checkbox2,'value')==1
        navio{1,2}=1;%vai se usar
lista de navios
    else
        navio{1,2}=0;%vai se usar
lista de navios indiferenciada
    end
    navio{1,3}=lista_navios;
navio{1,4}=lista_navios_indiferenciada
;
setappdata(handles.pushbutton1,'navio'
,navio)
 alarme=[]; %alarme é uma struct--
setappdata(handles.pushbutton1,'alarme
',alarme)
%%%% Estrutura dados GDH %%%%
gdh{1,1}=get(handles.checkbox17,'value
');
    gdh{1,2}{1,1}='validade';
gdh{1,2}{1,2}=datenum(get(handles.edit
26,'string'),'yyyy-mm-dd HH:MM:SS');
gdh{1,2}{1,3}=datenum(get(handles.edit
27,'string'),'yyyy-mm-dd HH:MM:SS');
    gdh{1,2}{2,1}='periodo horário';
gdh{1,2}{2,2}=get(handles.edit28,'stri
ng');
gdh{1,2}{2,3}=get(handles.edit29,'stri
ng');
setappdata(handles.pushbutton1,'gdh',g
dh)
%%% Estrutura dados METOC %%%%%%
metoc{1,1}=get(handles.checkbox6,'valu
e');
    m{1,1}='TWS';
m{1,2}=get(handles.checkbox9,'value');
m{1,3}=str2double(get(handles.edit15,'
string'));
m{1,4}=str2double(get(handles.edit16,'
string'));
    m{2,1}='TWD';
m{2,2}=get(handles.checkbox10,'value')
;
m{2,3}=str2double(get(handles.edit17,'
string'));

m{2,4}=str2double(get(handles.edit18,'
string'));
    m{3,1}='SWH';
m{3,2}=get(handles.checkbox11,'value')
;
m{3,3}=str2double(get(handles.edit19,'
string'));
m{3,4}=str2double(get(handles.edit20,'
string'));
    m{4,1}='WDir';
m{4,2}=get(handles.checkbox12,'value')
;
m{4,3}=str2double(get(handles.edit21,'
string'));
m{4,4}=str2double(get(handles.edit22,'
string'));
    metoc{1,2}=m;
setappdata(handles.pushbutton1,'metoc'
,metoc)
%%% Estrutura dados Poligono %%%%
    poligono=[];
setappdata(handles.pushbutton1,'poligo
no',poligono)
area{1,1}=get(handles.checkbox16,'valu
e');
    area{1,2}=poligono;
    if
get(handles.checkbox13,'value')==1
        area{1,3}=1;
    elseif
get(handles.checkbox14,'value')==1
        area{1,3}=2;
    else
        area{1,3}=3;
    end
area{1,4}=str2double(get(handles.edit2
4,'string'));
setappdata(handles.pushbutton1,'area',
area)
    if
get(handles.checkbox13,'value')==1
```

```
set(handles.checkbox14,'value',0)
set(handles.checkbox15,'value',0)
set(handles.edit24,'enable','off')
    else
set(handles.checkbox14,'value',1)
set(handles.checkbox15,'value',0)
set(handles.edit24,'enable','off')
    end
else
%%% Load %%%%%%%%%
dir_trabalho=getappdata(handles.pushbu
tton1,'dir_trabalho');
    pathname=[dir_trabalho
'\alarmes\']
    filename=varargin{1}
    if isnumeric(filename) &&
filename==0
    else
    %Ler ficheiro EXCEL
load([pathname filename])%carrega qeo
e cabecalho
        who
        alarme
        %%%actualizar informação do
painel GDH
set(handles.checkbox17,'value',alarme.
gdh{1,1})%actualização da codição
lógica no painel GDH
%%%% Estrutura dados GDH %%%%%
set(handles.edit26,'string',datestr(al
arme.gdh{1,2}{1,2},31),'Foregroundcolo
r','b')
set(handles.edit27,'string',datestr(al
arme.gdh{1,2}{1,3},31),'Foregroundcolo
r','b')
set(handles.edit28,'string',num2str(al
arme.gdh{1,2}{2,2}),'Foregroundcolor',
'b')
set(handles.edit29,'string',num2str(al
arme.gdh{1,2}{2,3}),'Foregroundcolor',
'b')
d1=get(handles.edit26,'string');
d1=datenum(d1,'yyyy-mm-dd HH:MM:SS');
d2=get(handles.edit27,'string');
        d2=datenum(d2,'yyyy-mm-dd
HH:MM:SS');
        dias=round(d2-d1+1);
set(handles.text33,'string',['Total em
dias: ' num2str(dias)])
        if
get(handles.checkbox17,'value')==1
set(handles.edit26,'enable','on')
set(handles.edit27,'enable','on')
set(handles.edit28,'enable','on')
set(handles.edit29,'enable','on')
        else
set(handles.checkbox17,'value',0)
set(handles.edit26,'enable','off')
set(handles.edit27,'enable','off')
set(handles.edit28,'enable','off')
set(handles.edit29,'enable','off')
        end
 if isfield(alarme,'prioridade')
set(handles.popupmenu2,'value',alarme.
prioridade,'foregroundcolor','b')
        else
        end
%actualizar informação do painel METOC
set(handles.checkbox6,'value',alarme.m
etoc{1,1})%actualização da codição
lógica no painel Poligono
set(handles.edit15,'string',num2str(al
arme.metoc{1,2}{1,3}),'Foregroundcolor
','b')
set(handles.edit16,'string',num2str(al
arme.metoc{1,2}{1,4}),'Foregroundcolor
','b')
set(handles.edit17,'string',num2str(al
arme.metoc{1,2}{2,3}),'Foregroundcolor
','b')
set(handles.edit18,'string',num2str(al
arme.metoc{1,2}{2,4}),'Foregroundcolor
','b')
set(handles.edit19,'string',num2str(al
arme.metoc{1,2}{3,3}),'Foregroundcolor
','b')
set(handles.edit20,'string',num2str(al
arme.metoc{1,2}{3,4}),'Foregroundcolor
','b')
set(handles.edit21,'string',num2str(al
arme.metoc{1,2}{4,3}),'Foregroundcolor
','b')
set(handles.edit22,'string',num2str(al
arme.metoc{1,2}{4,4}),'Foregroundcolor
','b')
set(handles.checkbox9,'value',alarme.m
etoc{1,2}{1,2})
set(handles.checkbox10,'value',alarme.
metoc{1,2}{2,2})
set(handles.checkbox11,'value',alarme.
metoc{1,2}{3,2})
set(handles.checkbox12,'value',alarme.
metoc{1,2}{4,2})
if get(handles.checkbox6,'value')==1
set(handles.checkbox9,'enable','on')
set(handles.edit15,'enable','on')
set(handles.edit16,'enable','on')
set(handles.checkbox10,'enable','on')
set(handles.edit17,'enable','on')
set(handles.edit18,'enable','on')
set(handles.checkbox11,'enable','on')
set(handles.edit19,'enable','on')
set(handles.edit20,'enable','on')
set(handles.checkbox12,'enable','on')
set(handles.edit21,'enable','on')
set(handles.edit22,'enable','on')
        else
set(handles.checkbox6,'value',0)
set(handles.checkbox9,'enable','off')
set(handles.edit15,'enable','off')
set(handles.edit16,'enable','off')
set(handles.checkbox10,'enable','off')
set(handles.edit17,'enable','off')
set(handles.edit18,'enable','off')
set(handles.checkbox11,'enable','off')
set(handles.edit19,'enable','off')
set(handles.edit20,'enable','off')
set(handles.checkbox12,'enable','off')
set(handles.edit21,'enable','off')
set(handles.edit22,'enable','off')
        end
        if
get(handles.checkbox9,'value')==1
set(handles.edit15,'enable','on')
set(handles.edit16,'enable','on')
        else
```

```
set(handles.edit15,'enable','off')
set(handles.edit16,'enable','off')
        end
        if
get(handles.checkbox10,'value')==1

set(handles.edit17,'enable','on')

set(handles.edit18,'enable','on')
        else

set(handles.edit17,'enable','off')

set(handles.edit18,'enable','off')
        end
        if
get(handles.checkbox11,'value')==1
set(handles.edit19,'enable','on')
set(handles.edit20,'enable','on')
        else
set(handles.edit19,'enable','off')
set(handles.edit20,'enable','off')
        end
        if
get(handles.checkbox12,'value')==1
set(handles.edit21,'enable','on')
set(handles.edit22,'enable','on')
        else
set(handles.edit21,'enable','off')
set(handles.edit22,'enable','off')
        end
%%%actualizar informação do painel
poligono
set(handles.checkbox16,'value',alarme.
area{1,1})%actualização da codição
lógica no painel Poligono
        if ~isempty(alarme.area{1,2})
set(handles.edit23,'string',alarme.are
a{1,2}{1,1},'Foregroundcolor','b')
set(handles.edit25,'string',alarme.are
a{1,2}{1,2},'Foregroundcolor','b')
        else
set(handles.edit23,'string','poligono'
,'Foregroundcolor','b')
set(handles.edit25,'string','sigla','F
oregroundcolor','b')
        end
set(handles.edit24,'string',num2str(al
arme.area{1,4}),'Foregroundcolor','b')
        if alarme.area{1,3}==1;
set(handles.checkbox13,'value',1)
set(handles.checkbox14,'value',0)
set(handles.checkbox15,'value',0)
set(handles.edit24,'enable','off')
        elseif alarme.area{1,3}==2;
set(handles.checkbox14,'value',1)
set(handles.checkbox13,'value',0)
set(handles.checkbox15,'value',0)
set(handles.edit24,'enable','off')
        else
set(handles.checkbox15,'value',1)
set(handles.checkbox14,'value',0)
set(handles.checkbox13,'value',0)
set(handles.edit24,'enable','on')
        end
set(handles.edit24,'string',num2str(al
arme.area{1,4}))
        poligono=alarme.area{1,2};
        if ~isempty(poligono)
            poligono
set(handles.text37,'string',['N° Total
de Pontos: '
num2str(numel(poligono{1,3}))])
            lat=poligono{1,5};
            lon=poligono{1,6};
            if isempty(lat) ||
isempty(lon)
                lat=poligono{1,3};
                lon=poligono{1,4};
            end
latmean=mean(lat);lonmean=mean(lon);
%limpar os poligonos do mapa-----
idx=findobj('tag','rect');
            if ~isempty(idx)
                delete(idx)
            end
idx=findobj('tag','pt');
            if ~isempty(idx)
                delete(idx)
            end
h=plotm(lat,lon,35,'r--
','Linewidth',1.5);set(h,'tag','rect')
;
h1=plotm(latmean,lonmean,'+k');set(h1,
'tag','pt');
        else
set(handles.text37,'string',['N° Total
de Pontos: ' num2str(0)])
        end
setappdata(handles.pushbutton1,'poligo
no',poligono)
        if
get(handles.checkbox13,'value')==1
set(handles.checkbox14,'value',0)
set(handles.checkbox15,'value',0)
set(handles.edit24,'enable','off')
        elseif
get(handles.checkbox14,'value')==1
set(handles.checkbox13,'value',0)
set(handles.checkbox15,'value',0)
set(handles.edit24,'enable','off')
        else
set(handles.checkbox15,'value',1)
set(handles.edit24,'enable','on')
        end
%--actualizar informação do painel
dados do alarmes---------
        if ~isempty(alarme.filename)
            alarme.filename
set(handles.edit7,'string',alarme.file
name,'Foregroundcolor','b')
        else
set(handles.edit7,'string','Dados não
disponiveis','Foregroundcolor','b')
        end
        if ~isempty(alarme.data)
set(handles.edit8,'string',datestr(ala
rme.data,'yyyy-mm-dd
HH:MM:SS'),'Foregroundcolor','b')
        else
set(handles.edit8,'string',['Data não
disponivel'],'Foregroundcolor','b')
        end
   if ~isempty(alarme.user)
set(handles.edit9,'string',alarme.user
,'Foregroundcolor','b')
```

```
        else
set(handles.edit9,'string',['User não
disponivel'],'Foregroundcolor','b')
        end
        if ~isempty(alarme.comments)

set(handles.edit10,'string',alarme.com
ments,'Foregroundcolor','b')
        else
set(handles.edit10,'string',['Comentár
ios não
disponiveis'],'Foregroundcolor','b')
        end
%----actualizar informação do painel
navio---------
set(handles.checkbox1,'value',alarme.a
rea{1,1})%actualização da codição
lógica no painel navio
lista_navios=alarme.navio{1,3}
        if  ~isempty(lista_navios)
  [n m]=size(lista_navios);
  l=1;%indice de linha individuo
            if l>n
                l=n;
            end
set(handles.uitable1,'userdata',l)
set(handles.edit1,'string',lista_navio
s{l,1})
set(handles.checkbox4,'value',lista_na
vios{l,2})
set(handles.edit3,'string',lista_navio
s{l,3})
set(handles.edit4,'string',lista_navio
s{l,4})
set(handles.checkbox5,'value',lista_na
vios{l,5})
set(handles.edit5,'string',lista_navio
s{l,6})
set(handles.edit6,'string',lista_navio
s{l,7})
        end
navio{1,1}=get(handles.checkbox1,'valu
e');
        if alarme.navio{1,2}==1
set(handles.checkbox2,'value',1);
set(handles.checkbox3,'value',0);
            CN={'MMSI','Use SOG','SOG
Min','SOG Max','Use COG','COG','COG
Variation'};
cw={120,100,80,80,100,80,120};
     foregroundColor = [1 1 1];
set(handles.uitable1,
'ForegroundColor', foregroundColor);
  backgroundColor = [.3 .6 .2; .2 .1
.4];
            set(handles.uitable1,
'BackgroundColor', backgroundColor);
set(handles.uitable1,'data',lista_navi
os,'ColumnName',CN,'rowstriping','on',
'ColumnWidth',cw)
set(handles.edit1,'enable','on')
set(handles.edit2,'enable','on')
set(handles.edit3,'enable','on')
set(handles.edit4,'enable','on')
set(handles.edit5,'enable','on')
set(handles.edit6,'enable','on')
set(handles.popupmenu1,'enable','off')
set(handles.pushbutton3,'enable','on')
set(handles.pushbutton4,'enable','on')
set(handles.pushbutton2,'enable','on')
lista_navios=alarme.navio{1,3}
setappdata(handles.pushbutton1,'lista_
navios', lista_navios);
lista_navios_indiferenciada=alarme.nav
io{1,4}
setappdata(handles.pushbutton1,'lista_
navios_indiferenciada',lista_navios_in
diferenciada)
set(handles.checkbox4,'value',alarme.n
avio{1,5}(1))%cehckbox do SOG
set(handles.edit3,'string',num2str(ala
rme.navio{1,5}(2)))%valor min SOG
set(handles.edit4,'string',num2str(ala
rme.navio{1,5}(3)))%valor max do SOG
set(handles.checkbox5,'value',alarme.n
avio{1,6}(1))%cehckbox do COG
set(handles.edit5,'string',num2str(ala
rme.navio{1,6}(2)))%valor min COG
set(handles.edit6,'string',num2str(ala
rme.navio{1,6}(3)))%valor max do COG
%set(handles.checkbox5,'value'
        else
set(handles.checkbox2,'value',0);
set(handles.checkbox3,'value',1);
        end
    end
end
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes analise_criar_alarmest2
wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are
returned to the command line.
function varargout =
analise_criar_alarmest2_OutputFcn(hObj
ect, eventdata, handles)
% varargout  cell array for returning
output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Get default command line output from
handles structure
varargout{1} = handles.output;

% --- Executes on slider movement.
function slider1_Callback(hObject,
eventdata, handles)
% hObject    handle to slider1 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hints: get(hObject,'Value') returns
position of slider
%        get(hObject,'Min') and
get(hObject,'Max') to determine range
of slider

% --- Executes during object creation,
after setting all properties.
```

```
function slider1_CreateFcn(hObject,
eventdata, handles)
% hObject    handle to slider1 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    empty - handles not
created until after all CreateFcns
called
% Hint: slider controls usually have a
light gray background.
if
isequal(get(hObject,'BackgroundColor')
,
get(0,'defaultUicontrolBackgroundColor
'))
set(hObject,'BackgroundColor',[.9 .9
.9]);
end

% --- Executes on button press in
checkbox1.
function checkbox1_Callback(hObject,
eventdata, handles)
% hObject    handle to checkbox1 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns
toggle state of checkbox1
if get(handles.checkbox1,'value')==1
set(handles.checkbox2,'value',1)
set(handles.checkbox3,'value',0)
set(handles.checkbox2,'enable','on')
set(handles.checkbox3,'enable','on')
set(handles.edit1,'enable','on')
set(handles.edit2,'enable','on')
set(handles.checkbox4,'enable','on')
set(handles.edit3,'enable','on')
set(handles.edit4,'enable','on')
set(handles.checkbox5,'enable','on')
set(handles.edit5,'enable','on')
set(handles.edit6,'enable','on')
set(handles.popupmenu1,'enable','on')
set(handles.popupmenu1,'enable','on')
set(handles.pushbutton1,'enable','on')
set(handles.pushbutton2,'enable','on')
set(handles.pushbutton3,'enable','on')
set(handles.pushbutton4,'enable','on')
else
set(handles.checkbox3,'value',0)
set(handles.checkbox2,'value',0)
set(handles.checkbox2,'enable','off')
set(handles.checkbox3,'enable','off')
set(handles.edit1,'enable','off')
set(handles.edit2,'enable','off')
set(handles.checkbox4,'enable','off')
set(handles.edit3,'enable','off')
set(handles.edit4,'enable','off')
set(handles.checkbox5,'enable','off')
set(handles.edit5,'enable','off')
set(handles.edit6,'enable','off')
set(handles.popupmenu1,'enable','off')
set(handles.popupmenu1,'enable','off')
set(handles.pushbutton1,'enable','off'
)
set(handles.pushbutton2,'enable','off'
)
set(handles.pushbutton3,'enable','off'
)
set(handles.pushbutton4,'enable','off'
)
end
%%%% Estrutura dados NAVIO %%%%%%
lista_navios=getappdata(handles.pushbu
tton1,'lista_navios');
lista_navios_indiferenciada=getappdata
(handles.pushbutton1,'lista_navios_ind
iferenciada');
navio{1,1}=get(handles.checkbox1,'valu
e');
if get(handles.checkbox2,'value')==1
    navio{1,2}=1;%vai se usar lista de
navios
else
    navio{1,2}=0;%vai se usar lista de
navios indiferenciada
end
navio{1,3}=lista_navios;
navio{1,4}=lista_navios_indiferenciada
;
setappdata(handles.pushbutton1,'navio'
,navio)
navio
navio{1,3}
navio{1,4}

% --- Executes on button press in
checkbox2.
function checkbox2_Callback(hObject,
eventdata, handles)
% hObject    handle to checkbox2 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns
toggle state of checkbox2
if get(handles.checkbox1,'value')==1
    if
get(handles.checkbox2,'value')==1
set(handles.checkbox3,'value',0)
set(handles.edit1,'enable','on')
set(handles.edit2,'enable','on')
set(handles.edit3,'enable','on')
set(handles.edit4,'enable','on')
set(handles.edit5,'enable','on')
set(handles.edit6,'enable','on')
set(handles.popupmenu1,'enable','off')
set(handles.pushbutton1,'enable','on')
set(handles.pushbutton2,'enable','on')
set(handles.pushbutton3,'enable','on')
set(handles.pushbutton4,'enable','on')
    else
set(handles.checkbox3,'value',1)
set(handles.edit1,'enable','off')
set(handles.edit2,'enable','off')
set(handles.edit3,'enable','on')
set(handles.edit4,'enable','on')
set(handles.edit5,'enable','on')
set(handles.edit6,'enable','on')

set(handles.popupmenu1,'enable','on')
```

```
set(handles.pushbutton1,'enable','off'
)
set(handles.pushbutton2,'enable','off'
)
set(handles.pushbutton3,'enable','off'
)
set(handles.pushbutton4,'enable','off'
)
    end
else
set(handles.checkbox2,'value',0)
end

% --- Executes on button press in
checkbox3.
function checkbox3_Callback(hObject,
eventdata, handles)
% hObject    handle to checkbox3 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns
toggle state of checkbox3
if get(handles.checkbox1,'value')==1
    if
get(handles.checkbox3,'value')==1
set(handles.checkbox2,'value',0)
set(handles.edit1,'enable','off')
set(handles.edit2,'enable','off')
set(handles.edit3,'enable','on')
set(handles.edit4,'enable','on')
set(handles.edit5,'enable','on')
set(handles.edit6,'enable','on')
set(handles.popupmenu1,'enable','on')
set(handles.pushbutton1,'enable','off'
)
set(handles.pushbutton2,'enable','off'
)
set(handles.pushbutton3,'enable','off'
)
set(handles.pushbutton4,'enable','off'
)
    else
set(handles.checkbox2,'value',1)
set(handles.edit1,'enable','on')
set(handles.edit2,'enable','on')
set(handles.edit3,'enable','on')
set(handles.edit4,'enable','on')
set(handles.edit5,'enable','on')
set(handles.edit6,'enable','on')
set(handles.popupmenu1,'enable','off')
set(handles.pushbutton1,'enable','on')
set(handles.pushbutton2,'enable','on')
set(handles.pushbutton3,'enable','on')
set(handles.pushbutton4,'enable','on')
    end
else
set(handles.checkbox3,'value',0)
end
function edit1_Callback(hObject,
eventdata, handles)
% hObject    handle to edit1 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hints: get(hObject,'String') returns
contents of edit1 as text
%
str2double(get(hObject,'String'))
returns contents of edit1 as a double

% --- Executes during object creation,
after setting all properties.
function edit1_CreateFcn(hObject,
eventdata, handles)
% hObject    handle to edit1 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    empty - handles not
created until after all CreateFcns
called
% Hint: edit controls usually have a
white background on Windows.
%       See ISPC and COMPUTER.
if ispc &&
isequal(get(hObject,'BackgroundColor')
,
get(0,'defaultUicontrolBackgroundColor
'))
set(hObject,'BackgroundColor','white')
;
end
function edit2_Callback(hObject,
eventdata, handles)
% hObject    handle to edit2 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hints: get(hObject,'String') returns
contents of edit2 as text
%
str2double(get(hObject,'String'))
returns contents of edit2 as a double

% --- Executes during object creation,
after setting all properties.
function edit2_CreateFcn(hObject,
eventdata, handles)
% hObject    handle to edit2 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    empty - handles not
created until after all CreateFcns
called
% Hint: edit controls usually have a
white background on Windows.
%       See ISPC and COMPUTER.
if ispc &&
isequal(get(hObject,'BackgroundColor')
,
get(0,'defaultUicontrolBackgroundColor
'))
set(hObject,'BackgroundColor','white')
;
end
function edit3_Callback(hObject,
eventdata, handles)
```

```
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hints: get(hObject,'String') returns
contents of edit3 as text
%
str2double(get(hObject,'String'))
returns contents of edit3 as a double

% --- Executes during object creation,
after setting all properties.
function edit3_CreateFcn(hObject,
eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    empty - handles not
created until after all CreateFcns
called
% Hint: edit controls usually have a
white background on Windows.
%       See ISPC and COMPUTER.
if ispc &&
isequal(get(hObject,'BackgroundColor')
,
get(0,'defaultUicontrolBackgroundColor
'))
set(hObject,'BackgroundColor','white')
;
end
function edit4_Callback(hObject,
eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hints: get(hObject,'String') returns
contents of edit4 as text
str2double(get(hObject,'String'))
returns contents of edit4 as a double

% --- Executes during object creation,
after setting all properties.
function edit4_CreateFcn(hObject,
eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    empty - handles not
created until after all CreateFcns
called
% Hint: edit controls usually have a
white background on Windows.
%       See ISPC and COMPUTER.
if ispc &&
isequal(get(hObject,'BackgroundColor')
,
get(0,'defaultUicontrolBackgroundColor
'))
set(hObject,'BackgroundColor','white')
;
end
function edit5_Callback(hObject,
eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hints: get(hObject,'String') returns
contents of edit5 as text
str2double(get(hObject,'String'))
returns contents of edit5 as a double

% --- Executes during object creation,
after setting all properties.
function edit5_CreateFcn(hObject,
eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    empty - handles not
created until after all CreateFcns
called
% Hint: edit controls usually have a
white background on Windows.
%       See ISPC and COMPUTER.
if ispc &&
isequal(get(hObject,'BackgroundColor')
,
get(0,'defaultUicontrolBackgroundColor
'))
set(hObject,'BackgroundColor','white')
;
end
function edit6_Callback(hObject,
eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hints: get(hObject,'String') returns
contents of edit6 as text
str2double(get(hObject,'String'))
returns contents of edit6 as a double

% --- Executes during object creation,
after setting all properties.
function edit6_CreateFcn(hObject,
eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    empty - handles not
created until after all CreateFcns
called
% Hint: edit controls usually have a
white background on Windows.
%       See ISPC and COMPUTER.
if ispc &&
isequal(get(hObject,'BackgroundColor')
,
get(0,'defaultUicontrolBackgroundColor
'))
set(hObject,'BackgroundColor','white')
;
end

% --- Executes on button press in
checkbox4.
function checkbox4_Callback(hObject,
eventdata, handles)
```

```
% hObject    handle to checkbox4 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns
toggle state of checkbox4

if get(handles.checkbox4,'value')==1
set(handles.edit3,'enable','on')
set(handles.edit4,'enable','on')
else
set(handles.edit3,'enable','off')
set(handles.edit4,'enable','off')
end

% --- Executes on button press in
checkbox5.
function checkbox5_Callback(hObject,
eventdata, handles)
% hObject    handle to checkbox5 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns
toggle state of checkbox5
if get(handles.checkbox5,'value')==1
set(handles.edit5,'enable','on')
set(handles.edit6,'enable','on')
else
set(handles.edit5,'enable','off')
set(handles.edit6,'enable','off')
end

% --- Executes on selection change in
popupmenu1.
function popupmenu1_Callback(hObject,
eventdata, handles)
% hObject    handle to popupmenu1 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hints: contents =
cellstr(get(hObject,'String')) returns
popupmenu1 contents as cell array
contents{get(hObject,'Value')} returns
selected item from popupmenu1

% --- Executes during object creation,
after setting all properties.
function popupmenu1_CreateFcn(hObject,
eventdata, handles)
% hObject    handle to popupmenu1 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    empty - handles not
created until after all CreateFcns
called
% Hint: popupmenu controls usually
have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc &&
isequal(get(hObject,'BackgroundColor')
,
get(0,'defaultUicontrolBackgroundColor
'))
set(hObject,'BackgroundColor','white')
;
end

% --- Executes on button press in
pushbutton1.
function pushbutton1_Callback(hObject,
eventdata, handles)
% hObject    handle to pushbutton1
(see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
%%%% Insert %%%%%%%%%%%%%%%%
%{if get(handles.checkbox4,'value')==1
   % if
str2num(get(handles.edit3,'string'))>s
tr2num(get(handles.edit4,'string'))
  %  d=msgbox('Velocidade mínima é
inferior à máxima!')
   % else
  %  end;
%elseif
get(handles.checkbox5,'value')==1
   % if
str2num(get(handles.edit5,'string'))>s
tr2num(get(handles.edit6,'string'))
    %d=msgbox('Velocidade mínima é
inferior à máxima!')
    %else%}
if get(handles.checkbox2,'value')==1
lista_navios=getappdata(handles.pushbu
tton1,'lista_navios');
    if isempty(lista_navios)
lista_navios{1,1}=str2double(get(handl
es.edit1,'string'));
if get(handles.checkbox4,'value')==1
            lista_navios{1,2}=1;
lista_navios{1,3}=str2double(get(handl
es.edit3,'string'));
lista_navios{1,4}=str2double(get(handl
es.edit4,'string'));
        else
            lista_navios{1,2}=0;
lista_navios{1,3}=str2double(get(handl
es.edit3,'string'));
lista_navios{1,4}=str2double(get(handl
es.edit4,'string'));
        end
        if
get(handles.checkbox5,'value')==1
            lista_navios{1,5}=1;
lista_navios{1,6}=str2double(get(handl
es.edit5,'string'));
lista_navios{1,7}=str2double(get(handl
es.edit6,'string'));
        else
            lista_navios{1,5}=0;
lista_navios{1,6}=str2double(get(handl
es.edit5,'string'));
lista_navios{1,7}=str2double(get(handl
es.edit6,'string'));
```

```
        end
    else
        [n m]=size(lista_navios);
lista_navios{n+1,1}=str2double(get(han
dles.edit1,'string'));
lista_navios{n+1,2}=get(handles.checkb
ox4,'value');
lista_navios{n+1,3}=str2double(get(han
dles.edit3,'string'));
lista_navios{n+1,4}=str2double(get(han
dles.edit4,'string'));
lista_navios{n+1,5}=get(handles.checkb
ox5,'value');
lista_navios{n+1,6}=str2double(get(han
dles.edit5,'string'));
lista_navios{n+1,7}=str2double(get(han
dles.edit6,'string'));
    end
%------  uitable -------------
setappdata(handles.pushbutton1,'lista_
navios', lista_navios);
    CN={'MMSI','Use SOG','SOG
Min','SOG Max','Use COG','COG','COG
Variation'};
    cw={120,100,80,80,100,80,120};
    foregroundColor = [1 1 1];
    set(handles.uitable1,
'ForegroundColor', foregroundColor);
    backgroundColor = [.3 .6 .2; .2 .1
.4];
    set(handles.uitable1,
'BackgroundColor', backgroundColor);
set(handles.uitable1,'data',lista_navi
os,'ColumnName',CN,'rowstriping','on',
'ColumnWidth',cw)
else
lista_navios_indiferenciada(1,1)=get(h
andles.popupmenu1,'value');
lista_navios_indiferenciada(1,2)=get(h
andles.checkbox4,'value');
lista_navios_indiferenciada(1,3)=str2d
ouble(get(handles.edit3,'string'));
lista_navios_indiferenciada(1,4)=str2d
ouble(get(handles.edit4,'string'));
lista_navios_indiferenciada(1,5)=get(h
andles.checkbox5,'value');
lista_navios_indiferenciada(1,6)=str2d
ouble(get(handles.edit5,'string'));
lista_navios_indiferenciada(1,7)=str2d
ouble(get(handles.edit5,'string'));
setappdata(handles.pushbutton1,'lista_
navios_indiferenciada',lista_navios_in
diferenciada)
    lista_navios_indiferenciada
end
%-- Estrutura dados NAVIO ------
lista_navios=getappdata(handles.pushbu
tton1,'lista_navios');
lista_navios_indiferenciada=getappdata
(handles.pushbutton1,'lista_navios_ind
iferenciada');
navio{1,1}=get(handles.checkbox1,'valu
e');
if get(handles.checkbox2,'value')==1
    navio{1,2}=1;%vai se usar lista de
navios
else
    navio{1,2}=0;%vai se usar lista de
navios indiferenciada
end
navio{1,3}=lista_navios;
navio{1,4}=lista_navios_indiferenciada
;
setappdata(handles.pushbutton1,'navio'
,navio)
%%%% Fim Insert %%%%%%%%%

% --- Executes on button press in
pushbutton2.
function pushbutton2_Callback(hObject,
eventdata, handles)
% hObject    handle to pushbutton2
(see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
lista_navios=getappdata(handles.pushbu
tton1,'lista_navios');[n
m]=size(lista_navios);
l=get(handles.uitable1,'userdata');
if ~isempty(lista_navios) && l<=n
    lista_navios(l,:)=[];
    CN={'MMSI','Use SOG','SOG
Min','SOG Max','Use COG','COG','COG
Variation'};
    cw={120,100,80,80,100,80,120};
    foregroundColor = [1 1 1];
    set(handles.uitable1,
'ForegroundColor', foregroundColor);
    backgroundColor = [.3 .6 .2; .2 .1
.4];
    set(handles.uitable1,
'BackgroundColor', backgroundColor);
set(handles.uitable1,'data',lista_navi
os,'ColumnName',CN,'rowstriping','on',
'ColumnWidth',cw)
setappdata(handles.pushbutton1,'lista_
navios', lista_navios);
end
%%%% Estrutura dados NAVIO %%%%%
lista_navios=getappdata(handles.pushbu
tton1,'lista_navios');
lista_navios_indiferenciada=getappdata
(handles.pushbutton1,'lista_navios_ind
iferenciada');
navio{1,1}=get(handles.checkbox1,'valu
e');
if get(handles.checkbox2,'value')==1
    navio{1,2}=1;%vai se usar lista de
navios
else
    navio{1,2}=0;%vai se usar lista de
navios indiferenciada
end
navio{1,3}=lista_navios;
navio{1,4}=lista_navios_indiferenciada
;
setappdata(handles.pushbutton1,'navio'
,navio)

% --- Executes on button press in
pushbutton3.
function pushbutton3_Callback(hObject,
eventdata, handles)
```

```
% hObject    handle to pushbutton3
(see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
lista_navios=getappdata(handles.pushbu
tton1,'lista_navios');
l=get(handles.uitable1,'userdata');
if ~isempty(lista_navios)
lista_navios{l,1}=str2double(get(handl
es.edit1,'string'));
lista_navios{l,2}=get(handles.checkbox
4,'value');
lista_navios{l,3}=str2double(get(handl
es.edit3,'string'));
lista_navios{l,4}=str2double(get(handl
es.edit4,'string'));
lista_navios{l,5}=get(handles.checkbox
5,'value');
lista_navios{l,6}=str2double(get(handl
es.edit5,'string'));
lista_navios{l,7}=str2double(get(handl
es.edit6,'string'));
    CN={'MMSI','Use SOG','SOG
Min','SOG Max','Use COG','COG','COG
Variation'};
    cw={120,100,80,80,100,80,120};
    foregroundColor = [1 1 1];
    set(handles.uitable1,
'ForegroundColor', foregroundColor);
    backgroundColor = [.3 .6 .2; .2 .1
.4];
    set(handles.uitable1,
'BackgroundColor', backgroundColor);
set(handles.uitable1,'data',lista_navi
os,'ColumnName',CN,'rowstriping','on',
'ColumnWidth',cw)
setappdata(handles.pushbutton1,'lista_
navios', lista_navios);
end
%%% Estrutura dados NAVIO %%%%%%
lista_navios=getappdata(handles.pushbu
tton1,'lista_navios');
lista_navios_indiferenciada=getappdata
(handles.pushbutton1,'lista_navios_ind
iferenciada');
navio{1,1}=get(handles.checkbox1,'valu
e');
if get(handles.checkbox2,'value')==1
    navio{1,2}=1;%vai se usar lista de
navios
else
    navio{1,2}=0;%vai se usar lista de
navios indiferenciada
end
navio{1,3}=lista_navios;
navio{1,4}=lista_navios_indiferenciada
;
setappdata(handles.pushbutton1,'navio'
,navio)

% --- Executes on button press in
pushbutton4.
function pushbutton4_Callback(hObject,
eventdata, handles)
% hObject    handle to pushbutton4
(see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
[h lista_mmsi]=select_mmsi
setappdata(handles.pushbutton1,'lista_
mmsi',lista_mmsi)
lista_navios=[]
if ~isempty(lista_mmsi)
    [n m]=size(lista_mmsi)
    for i =1:n
lista_navios{i,1}=lista_mmsi{i,1}
lista_navios{i,2}=get(handles.checkbox
4,'value');
lista_navios{i,3}=get(handles.edit3,'s
tring');
lista_navios{i,4}=get(handles.edit4,'s
tring');
lista_navios{i,5}=get(handles.checkbox
5,'value');
lista_navios{i,6}=get(handles.edit5,'s
tring');
lista_navios{i,7}=get(handles.edit6,'s
tring');
    end
end
lista_navios
%%%%%%%%%%  uitable %%%%%
setappdata(handles.pushbutton1,'lista_
navios', lista_navios);
CN={'MMSI','Use SOG','SOG Min','SOG
Max','Use COG','COG','COG Variation'};
cw={120,100,80,80,100,80,120};
foregroundColor = [1 1 1];
set(handles.uitable1,
'ForegroundColor', foregroundColor);
backgroundColor = [.3 .6 .2; .2 .1
.4];
set(handles.uitable1,
'BackgroundColor', backgroundColor);
set(handles.uitable1,'data',lista_navi
os,'ColumnName',CN,'rowstriping','on',
'ColumnWidth',cw)

% --- Executes on button press in
pushbutton5.
function pushbutton5_Callback(hObject,
eventdata, handles)
% hObject    handle to pushbutton5
(see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
%%%%% Load %%%%%
dir_trabalho=getappdata(handles.pushbu
tton1,'dir_trabalho');
[filename pathname] =
uigetfile({[dir_trabalho
'\alarmes\*.mat']},'');
if isnumeric(filename) && filename==0
else
    %Ler ficheiro EXCEL
    load([pathname filename])%carrega
qeo e cabecalho
    who
    alarme
%actualizar informação do painel GDH
```

```
set(handles.checkbox17,'value',alarme.
gdh{1,1})%actualização da codição
lógica no painel GDH
    %%%% Estrutura dados GDH %%%
set(handles.edit26,'string',datestr(al
arme.gdh{1,2}{1,2},31),'Foregroundcolo
r','b')
set(handles.edit27,'string',datestr(al
arme.gdh{1,2}{1,3},31),'Foregroundcolo
r','b')
set(handles.edit28,'string',num2str(al
arme.gdh{1,2}{2,2}),'Foregroundcolor',
'b')
set(handles.edit29,'string',num2str(al
arme.gdh{1,2}{2,3}),'Foregroundcolor',
'b')
d1=get(handles.edit26,'string');
    d1=datenum(d1,'yyyy-mm-dd
HH:MM:SS');
d2=get(handles.edit27,'string');
    d2=datenum(d2,'yyyy-mm-dd
HH:MM:SS');
    dias=round(d2-d1+1);
set(handles.text33,'string',['Total em
dias: ' num2str(dias)])
    if
get(handles.checkbox17,'value')==1
set(handles.edit26,'enable','on')
set(handles.edit27,'enable','on')
set(handles.edit28,'enable','on')
set(handles.edit29,'enable','on')
    else
set(handles.checkbox17,'value',0)
set(handles.edit26,'enable','off')
set(handles.edit27,'enable','off')
set(handles.edit28,'enable','off')
set(handles.edit29,'enable','off')
    end
%actualizar informação do painel METOC
set(handles.checkbox6,'value',alarme.m
etoc{1,1})%actualização da codição
lógica no painel Poligono
set(handles.edit15,'string',num2str(al
arme.metoc{1,2}{1,3}),'Foregroundcolor
','b')
set(handles.edit16,'string',num2str(al
arme.metoc{1,2}{1,4}),'Foregroundcolor
','b')
set(handles.edit17,'string',num2str(al
arme.metoc{1,2}{2,3}),'Foregroundcolor
','b')
set(handles.edit18,'string',num2str(al
arme.metoc{1,2}{2,4}),'Foregroundcolor
','b')
set(handles.edit19,'string',num2str(al
arme.metoc{1,2}{3,3}),'Foregroundcolor
','b')
set(handles.edit20,'string',num2str(al
arme.metoc{1,2}{3,4}),'Foregroundcolor
','b')
set(handles.edit21,'string',num2str(al
arme.metoc{1,2}{4,3}),'Foregroundcolor
','b')
set(handles.edit22,'string',num2str(al
arme.metoc{1,2}{4,4}),'Foregroundcolor
','b')
set(handles.checkbox9,'value',alarme.m
etoc{1,2}{1,2})
set(handles.checkbox10,'value',alarme.
metoc{1,2}{2,2})
set(handles.checkbox11,'value',alarme.
metoc{1,2}{3,2})
set(handles.checkbox12,'value',alarme.
metoc{1,2}{4,2})
    if
get(handles.checkbox6,'value')==1
set(handles.checkbox9,'enable','on')
set(handles.edit15,'enable','on')
set(handles.edit16,'enable','on')
set(handles.checkbox10,'enable','on')
set(handles.edit17,'enable','on')
set(handles.edit18,'enable','on')
set(handles.checkbox11,'enable','on')
set(handles.edit19,'enable','on')
set(handles.edit20,'enable','on')
set(handles.checkbox12,'enable','on')
set(handles.edit21,'enable','on')
set(handles.edit22,'enable','on')
    else
set(handles.checkbox6,'value',0)
set(handles.checkbox9,'enable','off')
set(handles.edit15,'enable','off')
set(handles.edit16,'enable','off')
set(handles.checkbox10,'enable','off')
set(handles.edit17,'enable','off')
set(handles.edit18,'enable','off')
set(handles.checkbox11,'enable','off')
set(handles.edit19,'enable','off')
set(handles.edit20,'enable','off')
set(handles.checkbox12,'enable','off')
set(handles.edit21,'enable','off')
set(handles.edit22,'enable','off')
    end
     if
get(handles.checkbox9,'value')==1
set(handles.edit15,'enable','on')
set(handles.edit16,'enable','on')
    else
set(handles.edit15,'enable','off')
set(handles.edit16,'enable','off')
    end
    if
get(handles.checkbox10,'value')==1
set(handles.edit17,'enable','on')
set(handles.edit18,'enable','on')
    else
set(handles.edit17,'enable','off')
set(handles.edit18,'enable','off')
    end
    if
get(handles.checkbox11,'value')==1
set(handles.edit19,'enable','on')
set(handles.edit20,'enable','on')
    else
set(handles.edit19,'enable','off')
set(handles.edit20,'enable','off')
    end
    if
get(handles.checkbox12,'value')==1
set(handles.edit21,'enable','on')
set(handles.edit22,'enable','on')
    else
set(handles.edit21,'enable','off')
set(handles.edit22,'enable','off')
    end
```

```
%actualizar informação do painel
poligono
set(handles.checkbox16,'value',alarme.
area{1,1})%actualização da codição
lógica no painel Poligono
    if ~isempty(alarme.area{1,2})
set(handles.edit23,'string',alarme.are
a{1,2}{1,1},'Foregroundcolor','b')
set(handles.edit25,'string',alarme.are
a{1,2}{1,2},'Foregroundcolor','b')
    else
set(handles.edit23,'string','poligono'
,'Foregroundcolor','b')
set(handles.edit25,'string','sigla','F
oregroundcolor','b')
    end
set(handles.edit24,'string',num2str(al
arme.area{1,4}),'Foregroundcolor','b')
    if alarme.area{1,3}==1;
set(handles.checkbox13,'value',1)
set(handles.checkbox14,'value',0)
set(handles.checkbox15,'value',0)
set(handles.edit24,'enable','off')
    elseif alarme.area{1,3}==2;
set(handles.checkbox14,'value',1)
set(handles.checkbox13,'value',0)
set(handles.checkbox15,'value',0)
set(handles.edit24,'enable','off')
    else
set(handles.checkbox15,'value',1)
set(handles.checkbox14,'value',0)
set(handles.checkbox13,'value',0)
set(handles.edit24,'enable','on')
    end
set(handles.edit24,'string',num2str(al
arme.area{1,4}))
    poligono=alarme.area{1,2};
    if ~isempty(poligono)
        poligono
set(handles.text37,'string',['N° Total
de Pontos: '
num2str(numel(poligono{1,3}))])
        lat=poligono{1,5};
        lon=poligono{1,6};
        if isempty(lat) ||
isempty(lon)
            lat=poligono{1,3};
            lon=poligono{1,4};
        end
latmean=mean(lat);lonmean=mean(lon);
%limpar os poligonos do mapa----
        idx=findobj('tag','rect');
        if ~isempty(idx)
            delete(idx)
        end
        idx=findobj('tag','pt');
        if ~isempty(idx)
            delete(idx)
        end
h=plotm(lat,lon,35,'r--
','Linewidth',1.5);set(h,'tag','rect')
;
h1=plotm(latmean,lonmean,'+k');set(h1,
'tag','pt');
    else
set(handles.text37,'string',['N° Total
de Pontos: ' num2str(0)])
    end
setappdata(handles.pushbutton1,'poligo
no',poligono)
    if
get(handles.checkbox13,'value')==1
set(handles.checkbox14,'value',0)
set(handles.checkbox15,'value',0)
set(handles.edit24,'enable','off')
    elseif
get(handles.checkbox14,'value')==1
set(handles.checkbox13,'value',0)
set(handles.checkbox15,'value',0)
set(handles.edit24,'enable','off')
    else
set(handles.checkbox15,'value',1)
set(handles.edit24,'enable','on')
    end
 %-----------actualizar informação do
painel dados do alarmes-----
     if ~isempty(alarme.filename)
         alarme.filename
set(handles.edit7,'string',alarme.file
name,'Foregroundcolor','b')
     else
set(handles.edit7,'string','Dados não
disponiveis','Foregroundcolor','b')
     end
     if ~isempty(alarme.data)
set(handles.edit8,'string',datestr(ala
rme.data,'yyyy-mm-dd
HH:MM:SS'),'Foregroundcolor','b')
     else
set(handles.edit8,'string',['Data não
disponivel'],'Foregroundcolor','b')
     end
     if ~isempty(alarme.user)
set(handles.edit9,'string',alarme.user
,'Foregroundcolor','b')
     else
set(handles.edit9,'string',['User não
disponivel'],'Foregroundcolor','b')
     end
     if ~isempty(alarme.comments)
set(handles.edit10,'string',alarme.com
ments,'Foregroundcolor','b')
     else
set(handles.edit10,'string',['Comentár
ios não
disponiveis'],'Foregroundcolor','b')
     end
%--actualizar informação do painel
navio---
set(handles.checkbox1,'value',alarme.a
rea{1,1})%actualização da codição
lógica no painel navio
lista_navios=alarme.navio{1,3};
    if  ~isempty(lista_navios)
        [n m]=size(lista_navios);
        l=1;%indice de linha individuo
        if l>n
            l=n;
        end
set(handles.uitable1,'userdata',l)
set(handles.edit1,'string',lista_navio
s{l,1})
set(handles.checkbox4,'value',lista_na
vios{l,2})
set(handles.edit3,'string',lista_navio
s{l,3})
```

```
set(handles.edit4,'string',lista_navio
s{1,4})
set(handles.checkbox5,'value',lista_na
vios{1,5})
set(handles.edit5,'string',lista_navio
s{1,6})
set(handles.edit6,'string',lista_navio
s{1,7})
    else
       CN={'MMSI','Use SOG','SOG
Min','SOG Max','Use COG','COG','COG
Variation'};
cw={120,100,80,80,100,80,120};
       foregroundColor = [1 1 1];
       set(handles.uitable1,
'ForegroundColor', foregroundColor);
       backgroundColor = [.3 .6 .2; .2
.1 .4];
       set(handles.uitable1,
'BackgroundColor', backgroundColor);
set(handles.uitable1,'data',lista_navi
os,'ColumnName',CN,'rowstriping','on',
'ColumnWidth',cw)
    end
navio{1,1}=get(handles.checkbox1,'valu
e');
    if alarme.navio{1,2}==1
set(handles.checkbox2,'value',1);
set(handles.checkbox3,'value',0);
       CN={'MMSI','Use SOG','SOG
Min','SOG Max','Use COG','COG','COG
Variation'};
cw={120,100,80,80,100,80,120};
       foregroundColor = [1 1 1];
       set(handles.uitable1,
'ForegroundColor', foregroundColor);
       backgroundColor = [.3 .6 .2; .2
.1 .4];
       set(handles.uitable1,
'BackgroundColor', backgroundColor);
set(handles.uitable1,'data',lista_navi
os,'ColumnName',CN,'rowstriping','on',
'ColumnWidth',cw)
set(handles.edit1,'enable','on')
set(handles.edit2,'enable','on')
set(handles.edit3,'enable','on')
set(handles.edit4,'enable','on')
set(handles.edit5,'enable','on')
set(handles.edit6,'enable','on')
set(handles.popupmenu1,'enable','off')
set(handles.pushbutton3,'enable','on')
set(handles.pushbutton4,'enable','on')
set(handles.pushbutton2,'enable','on')
lista_navios=alarme.navio{1,3};
setappdata(handles.pushbutton1,'lista_
navios', lista_navios);
lista_navios_indiferenciada=alarme.nav
io{1,4};
setappdata(handles.pushbutton1,'lista_
navios_indiferenciada',lista_navios_in
diferenciada)
    else
set(handles.checkbox2,'value',0);
set(handles.checkbox3,'value',1);
set(handles.popupmenu1,'value',alarme.
navio{1,4},'visible','on')
set(handles.edit1,'enable','off')
set(handles.edit2,'enable','off')
set(handles.popupmenu1,'enable','on')
set(handles.pushbutton3,'enable','off'
)
set(handles.pushbutton4,'enable','off'
)
set(handles.pushbutton2,'enable','off'
)
    end
    [n m]=size(alarme.navio);
    if m>4
set(handles.checkbox4,'value',alarme.n
avio{1,5}(1))%cehckbox do SOG
set(handles.edit3,'string',num2str(ala
rme.navio{1,5}(2)))%valor min SOG
set(handles.edit4,'string',num2str(ala
rme.navio{1,5}(3)))%valor max do SOG
set(handles.checkbox5,'value',alarme.n
avio{1,6}(1))%cehckbox do COG
set(handles.edit5,'string',num2str(ala
rme.navio{1,6}(2)))%valor min COG
set(handles.edit6,'string',num2str(ala
rme.navio{1,6}(3)))%valor max do COG
    end
%-atualizar informação da prioridade
    if isfield(alarme,'prioridade')
set(handles.popupmenu2,'value',alarme.
prioridade,'foregroundcolor','b')
    else
    end
setappdata(handles.pushbutton1,'lista_
navios', lista_navios);
end
%%% Fim Load %%
% --- Executes on button press in
pushbutton6.
function pushbutton6_Callback(hObject,
eventdata, handles)
% hObject    handle to pushbutton6
(see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
%%%% Gravar %%%%%
dir_trabalho=getappdata(handles.pushbu
tton1,'dir_trabalho');
%-----Gravar dados GDH----
gdh{1,1}=get(handles.checkbox17,'value
');
gdh{1,2}{1,1}='validade';
gdh{1,2}{1,2}=datenum(strtrim(get(hand
les.edit26,'string')),'yyyy-mm-dd
HH:MM:SS');
gdh{1,2}{1,3}=datenum(strtrim(get(hand
les.edit27,'string')),'yyyy-mm-dd
HH:MM:SS');
gdh{1,2}{2,1}='periodo horário';
gdh{1,2}{2,2}=get(handles.edit28,'stri
ng');
gdh{1,2}{2,3}=get(handles.edit29,'stri
ng');
setappdata(handles.pushbutton1,'gdh',g
dh)
%----Gravar dados Navio-------
lista_navios=getappdata(handles.pushbu
tton1,'lista_navios');
```

```
lista_navios_indiferenciada=getappdata
(handles.pushbutton1,'lista_navios_ind
iferenciada');
navio{1,1}=get(handles.checkbox1,'valu
e');
if get(handles.checkbox2,'value')==1
    navio{1,2}=1;%vai se usar lista de
navios
else
    navio{1,2}=0;%vai se usar lista de
navios indiferenciada
end
navio{1,3}=lista_navios;
navio{1,4}=get(handles.popupmenu1,'val
ue');%tipo de navio
navio{1,5}=[get(handles.checkbox4,'val
ue')
str2double(get(handles.edit3,'string')
)
str2double(get(handles.edit4,'string')
)];%SOG
navio{1,6}=[get(handles.checkbox5,'val
ue')
str2double(get(handles.edit5,'string')
)
str2double(get(handles.edit6,'string')
)];%COG
setappdata(handles.pushbutton1,'navio'
,navio)
%---Gravar dados Metoc-----
metoc{1,1}=get(handles.checkbox6,'valu
e');
m{1,1}='TWS';
m{1,2}=get(handles.checkbox9,'value');
m{1,3}=str2double(get(handles.edit15,'
string'));
m{1,4}=str2double(get(handles.edit16,'
string'));
m{2,1}='TWD';
m{2,2}=get(handles.checkbox10,'value')
;
m{2,3}=str2double(get(handles.edit17,'
string'));
m{2,4}=str2double(get(handles.edit18,'
string'));
m{3,1}='SWH';
m{3,2}=get(handles.checkbox11,'value')
;
m{3,3}=str2double(get(handles.edit19,'
string'));
m{3,4}=str2double(get(handles.edit20,'
string'));
m{4,1}='WDir';
m{4,2}=get(handles.checkbox12,'value')
;
m{4,3}=str2double(get(handles.edit21,'
string'));
m{4,4}=str2double(get(handles.edit22,'
string'));
metoc{1,2}=m;
setappdata(handles.pushbutton1,'metoc'
,metoc)
%-----Gravar dados Poligono------
poligono=getappdata(handles.pushbutton
1,'poligono');
area{1,1}=get(handles.checkbox16,'valu
e');
area{1,2}=poligono;
if get(handles.checkbox13,'value')==1
    area{1,3}=1;
elseif
get(handles.checkbox14,'value')==1
    area{1,3}=2;
else
    area{1,3}=3;
end
area{1,2}{1,1}=get(handles.edit23,'str
ing')
area{1,2}{1,2}=get(handles.edit25,'str
ing')
area{1,4}=str2double(get(handles.edit2
4,'string'));
setappdata(handles.pushbutton1,'area',
area)
%--- Struct alarme----------
alarme.gdh=gdh
alarme.navio=navio
alarme.metoc=metoc
alarme.area=area
alarme.user=get(handles.edit9,'string'
)
alarme.filename=get(handles.edit7,'str
ing')
alarme.data=datenum(get(handles.edit8,
'string'),'yyyy-mm-dd HH:MM:SS')
alarme.dados=[]
alarme.comments=get(handles.edit10,'st
ring')
alarme.prioridade=get(handles.popupmen
u2,'value')
dir_trabalho=getappdata(handles.pushbu
tton1,'dir_trabalho');
if exist([dir_trabalho '\alarmes\'
],'dir')
else
    mkdir([dir_trabalho '\alarmes\' ])
end
pathname=[dir_trabalho '\alarmes\' ];
filename=get(handles.edit7,'string')
if exist([pathname filename],'file')
 %Perguntar se deseja actualizar o
alarme-----
    % Construct a questdlg with three
options
    choice = questdlg('Já existe um
alarme com esse nome. Pretende
continuar?', ...
 'Menu de Gravação', ...
'Sim','Não','Cancelar','Cancelar');
    % Handle response
    dessert=0;
    switch choice
        case 'Sim'
            disp([choice ' coming
right up.'])
            dessert = 1;
            save([pathname
filename],'alarme')
        case 'Não'
            disp('I''ll bring you your
check.')
            dessert = 0;
        case 'Cancelar'
    end
else
%Perguntar se deseja mesmo gravar-
```

```
    % Construct a questdlg with three
options
    choice = questdlg('Confirma
intenção em gravar o alarme?', ...
        'Menu de Gravação', ...
        'Sim','Não','Não');
    % Handle response
    dessert=0;
    switch choice
        case 'Sim'
            disp([choice ' coming
right up.'])
            dessert = 1;
            save([pathname
filename],'alarme')
        case 'Não'
            disp('I''ll bring you your
check.')
            dessert = 0;
    end
end
%%% Fim gravar %%%
% --- Executes on button press in
pushbutton7.
function pushbutton7_Callback(hObject,
eventdata, handles)
% hObject    handle to pushbutton7
(see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
close
function edit7_Callback(hObject,
eventdata, handles)
% hObject    handle to edit7 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hints: get(hObject,'String') returns
contents of edit7 as text
%
str2double(get(hObject,'String'))
returns contents of edit7 as a double
set(handles.edit7,'Foregroundcolor','b
')

% --- Executes during object creation,
after setting all properties.
function edit7_CreateFcn(hObject,
eventdata, handles)
% hObject    handle to edit7 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    empty - handles not
created until after all CreateFcns
called
% Hint: edit controls usually have a
white background on Windows.
%       See ISPC and COMPUTER.
if ispc &&
isequal(get(hObject,'BackgroundColor')
,
get(0,'defaultUicontrolBackgroundColor
'))
set(hObject,'BackgroundColor','white')
;
end
function edit8_Callback(hObject,
eventdata, handles)
% hObject    handle to edit8 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hints: get(hObject,'String') returns
contents of edit8 as text
%
str2double(get(hObject,'String'))
returns contents of edit8 as a double
set(handles.edit8,'Foregroundcolor','b
')

% --- Executes during object creation,
after setting all properties.
function edit8_CreateFcn(hObject,
eventdata, handles)
% hObject    handle to edit8 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    empty - handles not
created until after all CreateFcns
called
% Hint: edit controls usually have a
white background on Windows.
%       See ISPC and COMPUTER.
if ispc &&
isequal(get(hObject,'BackgroundColor')
,
get(0,'defaultUicontrolBackgroundColor
'))
set(hObject,'BackgroundColor','white')
;
end
function edit9_Callback(hObject,
eventdata, handles)
% hObject    handle to edit9 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hints: get(hObject,'String') returns
contents of edit9 as text
str2double(get(hObject,'String'))
returns contents of edit9 as a double
set(handles.edit9,'Foregroundcolor','b
')

% --- Executes during object creation,
after setting all properties.
function edit9_CreateFcn(hObject,
eventdata, handles)
% hObject    handle to edit9 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    empty - handles not
created until after all CreateFcns
called
```

```
% Hint: edit controls usually have a
white background on Windows.
%       See ISPC and COMPUTER.
if ispc &&
isequal(get(hObject,'BackgroundColor')
,
get(0,'defaultUicontrolBackgroundColor
'))
set(hObject,'BackgroundColor','white')
;
end
function edit10_Callback(hObject,
eventdata, handles)
% hObject    handle to edit10 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hints: get(hObject,'String') returns
contents of edit10 as text
%
str2double(get(hObject,'String'))
returns contents of edit10 as a double
set(handles.edit10,'Foregroundcolor','
b')

% --- Executes during object creation,
after setting all properties.
function edit10_CreateFcn(hObject,
eventdata, handles)
% hObject    handle to edit10 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    empty - handles not
created until after all CreateFcns
called
% Hint: edit controls usually have a
white background on Windows.
%       See ISPC and COMPUTER.
if ispc &&
isequal(get(hObject,'BackgroundColor')
,
get(0,'defaultUicontrolBackgroundColor
'))
set(hObject,'BackgroundColor','white')
;
end

% --- Executes on button press in
checkbox6.
function checkbox6_Callback(hObject,
eventdata, handles)
% hObject    handle to checkbox6 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns
toggle state of checkbox6
if get(handles.checkbox6,'value')==1
set(handles.checkbox9,'value',1)
set(handles.checkbox10,'value',1)
set(handles.checkbox11,'value',1)
set(handles.checkbox12,'value',1)
set(handles.checkbox9,'enable','on')
set(handles.edit15,'enable','on')
set(handles.edit16,'enable','on')
set(handles.checkbox10,'enable','on')
set(handles.edit17,'enable','on')
set(handles.edit18,'enable','on')
set(handles.checkbox11,'enable','on')
set(handles.edit19,'enable','on')
set(handles.edit20,'enable','on')
set(handles.checkbox12,'enable','on')
set(handles.edit21,'enable','on')
set(handles.edit22,'enable','on')
else
set(handles.checkbox6,'value',0)
set(handles.checkbox9,'enable','off')
set(handles.edit15,'enable','off')
set(handles.edit16,'enable','off')
set(handles.checkbox10,'enable','off')
set(handles.edit17,'enable','off')
set(handles.edit18,'enable','off')
set(handles.checkbox11,'enable','off')
set(handles.edit19,'enable','off')
set(handles.edit20,'enable','off')
set(handles.checkbox12,'enable','off')
set(handles.edit21,'enable','off')
set(handles.edit22,'enable','off')
end
metoc{1,1}=get(handles.checkbox6,'valu
e');
m{1,1}='TWS';
m{1,2}=get(handles.checkbox9,'value');
m{1,3}=str2double(get(handles.edit15,'
string'));
m{1,4}=str2double(get(handles.edit16,'
string'));
m{2,1}='TWD';
m{2,2}=get(handles.checkbox10,'value')
;
m{2,3}=str2double(get(handles.edit17,'
string'));
m{2,4}=str2double(get(handles.edit18,'
string'));
m{3,1}='SWH';
m{3,2}=get(handles.checkbox11,'value')
;
m{3,3}=str2double(get(handles.edit19,'
string'));
m{3,4}=str2double(get(handles.edit20,'
string'));
m{4,1}='WDir';
m{4,2}=get(handles.checkbox12,'value')
;
m{4,3}=str2double(get(handles.edit21,'
string'));
m{4,4}=str2double(get(handles.edit22,'
string'));
metoc{1,2}=m;
setappdata(handles.pushbutton1,'metoc'
,metoc)
metoc
metoc{1,2}
function edit15_Callback(hObject,
eventdata, handles)
% hObject    handle to edit15 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns
contents of edit15 as text
%
str2double(get(hObject,'String'))
returns contents of edit15 as a double

% --- Executes during object creation,
after setting all properties.
function edit15_CreateFcn(hObject,
eventdata, handles)
% hObject    handle to edit15 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    empty - handles not
created until after all CreateFcns
called
% Hint: edit controls usually have a
white background on Windows.
%       See ISPC and COMPUTER.
if ispc &&
isequal(get(hObject,'BackgroundColor')
,
get(0,'defaultUicontrolBackgroundColor
'))
set(hObject,'BackgroundColor','white')
;
end
function edit16_Callback(hObject,
eventdata, handles)
% hObject    handle to edit16 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hints: get(hObject,'String') returns
contents of edit16 as text
%
str2double(get(hObject,'String'))
returns contents of edit16 as a double

% --- Executes during object creation,
after setting all properties.
function edit16_CreateFcn(hObject,
eventdata, handles)
% hObject    handle to edit16 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    empty - handles not
created until after all CreateFcns
called
% Hint: edit controls usually have a
white background on Windows.
%       See ISPC and COMPUTER.
if ispc &&
isequal(get(hObject,'BackgroundColor')
,
get(0,'defaultUicontrolBackgroundColor
'))
set(hObject,'BackgroundColor','white')
;
end
function edit17_Callback(hObject,
eventdata, handles)
% hObject    handle to edit17 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hints: get(hObject,'String') returns
contents of edit17 as text
%
str2double(get(hObject,'String'))
returns contents of edit17 as a double

% --- Executes during object creation,
after setting all properties.
function edit17_CreateFcn(hObject,
eventdata, handles)
% hObject    handle to edit17 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    empty - handles not
created until after all CreateFcns
called
% Hint: edit controls usually have a
white background on Windows.
%       See ISPC and COMPUTER.
if ispc &&
isequal(get(hObject,'BackgroundColor')
,
get(0,'defaultUicontrolBackgroundColor
'))
set(hObject,'BackgroundColor','white')
;
end
function edit18_Callback(hObject,
eventdata, handles)
% hObject    handle to edit18 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hints: get(hObject,'String') returns
contents of edit18 as text
str2double(get(hObject,'String'))
returns contents of edit18 as a double

% --- Executes during object creation,
after setting all properties.
function edit18_CreateFcn(hObject,
eventdata, handles)
% hObject    handle to edit18 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    empty - handles not
created until after all CreateFcns
called
% Hint: edit controls usually have a
white background on Windows.
%       See ISPC and COMPUTER.
if ispc &&
isequal(get(hObject,'BackgroundColor')
,
get(0,'defaultUicontrolBackgroundColor
'))
set(hObject,'BackgroundColor','white')
;
end
```

```
% --- Executes on button press in
checkbox9.
function checkbox9_Callback(hObject,
eventdata, handles)
% hObject    handle to checkbox9 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns
toggle state of checkbox9
if get(handles.checkbox9,'value')==1
set(handles.edit15,'enable','on')
set(handles.edit16,'enable','on')
else
set(handles.edit15,'enable','off')
    set(handles.edit16,'enable','off')
end

% --- Executes on button press in
checkbox10.
function checkbox10_Callback(hObject,
eventdata, handles)
% hObject    handle to checkbox10 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns
toggle state of checkbox10
if get(handles.checkbox10,'value')==1
set(handles.edit17,'enable','on')
set(handles.edit18,'enable','on')
else
set(handles.edit17,'enable','off')
set(handles.edit18,'enable','off')
end
function edit19_Callback(hObject,
eventdata, handles)
% hObject    handle to edit19 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hints: get(hObject,'String') returns
contents of edit19 as text
str2double(get(hObject,'String'))
returns contents of edit19 as a double

% --- Executes during object creation,
after setting all properties.
function edit19_CreateFcn(hObject,
eventdata, handles)
% hObject    handle to edit19 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    empty - handles not
created until after all CreateFcns
called
% Hint: edit controls usually have a
white background on Windows.
%       See ISPC and COMPUTER.
if ispc &&
isequal(get(hObject,'BackgroundColor')
,
get(0,'defaultUicontrolBackgroundColor
'))
set(hObject,'BackgroundColor','white')
;
end
function edit20_Callback(hObject,
eventdata, handles)
% hObject    handle to edit20 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hints: get(hObject,'String') returns
contents of edit20 as text
%
str2double(get(hObject,'String'))
returns contents of edit20 as a double

% --- Executes during object creation,
after setting all properties.
function edit20_CreateFcn(hObject,
eventdata, handles)
% hObject    handle to edit20 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    empty - handles not
created until after all CreateFcns
called
% Hint: edit controls usually have a
white background on Windows.
%       See ISPC and COMPUTER.
if ispc &&
isequal(get(hObject,'BackgroundColor')
,
get(0,'defaultUicontrolBackgroundColor
'))
set(hObject,'BackgroundColor','white')
;
end
function edit21_Callback(hObject,
eventdata, handles)
% hObject    handle to edit21 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hints: get(hObject,'String') returns
contents of edit21 as text
str2double(get(hObject,'String'))
returns contents of edit21 as a double

% --- Executes during object creation,
after setting all properties.
function edit21_CreateFcn(hObject,
eventdata, handles)
% hObject  handle to edit21 (see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    empty - handles not
created until after all CreateFcns
called
% Hint: edit controls usually have a
white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc &&
isequal(get(hObject,'BackgroundColor')
,
get(0,'defaultUicontrolBackgroundColor
'))
set(hObject,'BackgroundColor','white')
;
end
function edit22_Callback(hObject,
eventdata, handles)
% hObject    handle to edit22 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hints: get(hObject,'String') returns
contents of edit22 as text
str2double(get(hObject,'String'))
returns contents of edit22 as a double

% --- Executes during object creation,
after setting all properties.
function edit22_CreateFcn(hObject,
eventdata, handles)
% hObject  handle to edit22 (see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    empty - handles not
created until after all CreateFcns
called
% Hint: edit controls usually have a
white background on Windows.
%       See ISPC and COMPUTER.
if ispc &&
isequal(get(hObject,'BackgroundColor')
,
get(0,'defaultUicontrolBackgroundColor
'))
set(hObject,'BackgroundColor','white')
;
end

% --- Executes on button press in
checkbox11.
function checkbox11_Callback(hObject,
eventdata, handles)
% hObject    handle to checkbox11 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns
toggle state of checkbox11

if get(handles.checkbox11,'value')==1
set(handles.edit19,'enable','on')
set(handles.edit20,'enable','on')
else
set(handles.edit19,'enable','off')
set(handles.edit20,'enable','off')
end

% --- Executes on button press in
checkbox12.
function checkbox12_Callback(hObject,
eventdata, handles)
% hObject    handle to checkbox12 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns
toggle state of checkbox12
if get(handles.checkbox12,'value')==1
set(handles.edit21,'enable','on')
set(handles.edit22,'enable','on')
else
set(handles.edit21,'enable','off')
set(handles.edit22,'enable','off')
end

% --- Executes on button press in
checkbox13.
function checkbox13_Callback(hObject,
eventdata, handles)
% hObject    handle to checkbox13 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns
toggle state of checkbox13
if get(handles.checkbox13,'value')==1
set(handles.checkbox14,'value',0)
set(handles.checkbox15,'value',0)
set(handles.edit24,'enable','off')
else
set(handles.checkbox14,'value',1)
set(handles.checkbox15,'value',0)
set(handles.edit24,'enable','off')
end
poligono=getappdata(handles.pushbutton
1,'poligono');
area{1,1}=get(handles.checkbox16,'valu
e');
area{1,2}=poligono;
if get(handles.checkbox13,'value')==1
    area{1,3}=1;
elseif
get(handles.checkbox14,'value')==1
    area{1,3}=2;
else
    area{1,3}=3;
end
area{1,4}=str2double(get(handles.edit2
4,'string'));
setappdata(handles.pushbutton1,'area',
area)

% --- Executes on button press in
checkbox14.
function checkbox14_Callback(hObject,
eventdata, handles)
% hObject handle to checkbox14 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns
toggle state of checkbox14
if get(handles.checkbox14,'value')==1
```

```
set(handles.checkbox13,'value',0)
set(handles.checkbox15,'value',0)
set(handles.edit24,'enable','off')
else
set(handles.checkbox13,'value',1)
set(handles.checkbox15,'value',0)
set(handles.edit24,'enable','off')
end
poligono=getappdata(handles.pushbutton
1,'poligono');
area{1,1}=get(handles.checkbox16,'valu
e');
area{1,2}=poligono;
if get(handles.checkbox13,'value')==1
    area{1,3}=1;
elseif
get(handles.checkbox14,'value')==1
    area{1,3}=2;
else
    area{1,3}=3;
end
area{1,4}=str2double(get(handles.edit2
4,'string'));
setappdata(handles.pushbutton1,'area',
area)

% --- Executes on button press in
checkbox15.
function checkbox15_Callback(hObject,
eventdata, handles)
% hObject    handle to checkbox15 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns
toggle state of checkbox15
if get(handles.checkbox15,'value')==1
set(handles.checkbox13,'value',0)
set(handles.checkbox14,'value',0)
set(handles.edit24,'enable','on')
else
set(handles.checkbox13,'value',1)
set(handles.checkbox15,'value',0)
set(handles.edit24,'enable','on')
end
poligono=getappdata(handles.pushbutton
1,'poligono');
area{1,1}=get(handles.checkbox16,'valu
e');
area{1,2}=poligono;
if get(handles.checkbox13,'value')==1
    area{1,3}=1;
elseif
get(handles.checkbox14,'value')==1
    area{1,3}=2;
else
    area{1,3}=3;
end
area{1,4}=str2double(get(handles.edit2
4,'string'));
setappdata(handles.pushbutton1,'area',
area)
function edit23_Callback(hObject,
eventdata, handles)
% hObject  handle to edit23 (see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hints: get(hObject,'String') returns
contents of edit23 as text
str2double(get(hObject,'String'))
returns contents of edit23 as a double
set(handles.edit23,'Foregroundcolor','
b')
poligono=getappdata(handles.pushbutton
1,'poligono');
area{1,1}=get(handles.checkbox16,'valu
e');
area{1,2}=poligono;
if get(handles.checkbox13,'value')==1
    area{1,3}=1;
elseif
get(handles.checkbox14,'value')==1
    area{1,3}=2;
else
    area{1,3}=3;
end
area{1,4}=str2double(get(handles.edit2
4,'string'));
setappdata(handles.pushbutton1,'area',
area)

% --- Executes during object creation,
after setting all properties.
function edit23_CreateFcn(hObject,
eventdata, handles)
% hObject  handle to edit23 (see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    empty - handles not
created until after all CreateFcns
called
% Hint: edit controls usually have a
white background on Windows.
%       See ISPC and COMPUTER.
if ispc &&
isequal(get(hObject,'BackgroundColor')
,
get(0,'defaultUicontrolBackgroundColor
'))
set(hObject,'BackgroundColor','white')
;
end
function edit24_Callback(hObject,
eventdata, handles)
% hObject  handle to edit24 (see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hints: get(hObject,'String') returns
contents of edit24 as text
str2double(get(hObject,'String'))
returns contents of edit24 as a double

% --- Executes during object creation,
after setting all properties.
function edit24_CreateFcn(hObject,
eventdata, handles)
% hObject  handle to edit24 (see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
```

```
% handles    empty - handles not
created until after all CreateFcns
called
% Hint: edit controls usually have a
white background on Windows.
%       See ISPC and COMPUTER.
if ispc &&
isequal(get(hObject,'BackgroundColor')
,
get(0,'defaultUicontrolBackgroundColor
'))
set(hObject,'BackgroundColor','white')
;
end

% --- Executes on button press in
checkbox16.
function checkbox16_Callback(hObject,
eventdata, handles)
% hObject    handle to checkbox16 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns
toggle state of checkbox16
if get(handles.checkbox16,'value')==1
set(handles.checkbox13,'enable','on')
set(handles.checkbox14,'enable','on')
set(handles.checkbox15,'enable','on')
set(handles.edit23,'enable','on')
set(handles.edit24,'enable','on')
set(handles.edit25,'enable','on')
set(handles.pushbutton8,'enable','on')
set(handles.pushbutton9,'enable','on')
set(handles.pushbutton10,'enable','on'
)
else
set(handles.checkbox16,'value',0)
set(handles.checkbox13,'enable','off')
set(handles.checkbox14,'enable','off')
set(handles.checkbox15,'enable','off')
set(handles.edit23,'enable','off')
set(handles.edit24,'enable','off')
set(handles.edit25,'enable','off')
set(handles.pushbutton8,'enable','off'
)
set(handles.pushbutton9,'enable','off'
)
set(handles.pushbutton10,'enable','off
')
end
poligono=getappdata(handles.pushbutton
1,'poligono');
area{1,1}=get(handles.checkbox16,'valu
e');
area{1,2}=poligono;
if get(handles.checkbox13,'value')==1
    area{1,3}=1;
elseif
get(handles.checkbox14,'value')==1
    area{1,3}=2;
else
    area{1,3}=3;
end
area{1,4}=str2double(get(handles.edit2
4,'string'));
setappdata(handles.pushbutton1,'area',
area)
area
area{1,2}
function edit25_Callback(hObject,
eventdata, handles)
% hObject  handle to edit25 (see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hints: get(hObject,'String') returns
contents of edit25 as text
str2double(get(hObject,'String'))
returns contents of edit25 as a double
set(handles.edit25,'Foregroundcolor','
b')
poligono=getappdata(handles.pushbutton
1,'poligono');
area{1,1}=get(handles.checkbox16,'valu
e');
area{1,2}=poligono;
if get(handles.checkbox13,'value')==1
    area{1,3}=1;
elseif
get(handles.checkbox14,'value')==1
    area{1,3}=2;
else
    area{1,3}=3;
end
area{1,4}=str2double(get(handles.edit2
4,'string'));
setappdata(handles.pushbutton1,'area',
area)

% --- Executes during object creation,
after setting all properties.
function edit25_CreateFcn(hObject,
eventdata, handles)
% hObject  handle to edit25 (see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    empty - handles not
created until after all CreateFcns
called
% Hint: edit controls usually have a
white background on Windows.
%       See ISPC and COMPUTER.
if ispc &&
isequal(get(hObject,'BackgroundColor')
,
get(0,'defaultUicontrolBackgroundColor
'))
set(hObject,'BackgroundColor','white')
;
end

% --- Executes on button press in
pushbutton8.
function pushbutton8_Callback(hObject,
eventdata, handles)
% hObject handle to pushbutton8 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
%%%%%% Rectângulo %%%%%%%%
```

```
zoom off
pan off
poligono=[];
setappdata(handles.pushbutton1,'poligo
no',poligono)
%limpar lisbox
idx=findobj('tag','rect');
if ~isempty(idx)
    delete(idx)
end
idx=findobj('tag','rmp');
if ~isempty(idx)
    delete(idx)
end
idx=findobj('tag','pos');
if ~isempty(idx)
    delete(idx)
end
%posicoes=getappdata(handles.pushbutto
n1,'posicoes');
%[l lixo]=size(posicoes);
%{
l=getappdata(handles.pushbutton1,'n_po
s');
for i=1:l
    idx=findobj('tag',num2str(i));
    if ~isempty(idx)
        delete(idx)
    end
end
setappdata(handles.pushbutton1,'n_pos'
,0)
%}
idx=findobj('tag','pt');
if ~isempty(idx)
    delete(idx)
end
k = waitforbuttonpress;
point1 = gcpmap;    % button down
detected
finalRect = rbbox;                   %
return figure units
point2 = gcpmap;    % button up
detected
point1 = point1(1,1:2);              %
extract x and y
point2 = point2(1,1:2);
p1 = min(point1,point2);             %
calculate locations
offset = abs(point1-point2);         %
and dimensions
x = [p1(1) p1(1)+offset(1)
p1(1)+offset(1) p1(1) p1(1)];
y = [p1(2) p1(2) p1(2)+offset(2)
p1(2)+offset(2) p1(2)];
hold on
h=plotm(x,y,15,'r--') ;
set(h,'tag','rect');
[latmean,lonmean] = meanm(x(1,1:end-
1),y(1,1:end-1));
h1=plotm(latmean,lonmean,27,'+k');set(
h1,'tag','pt','markersize',12);
lat_max=max(x);lat_min=min(x);
lon_max=max(y);lon_min=min(y);
lat(1,1)=lat_max;
lat(2,1)=lat_max;
lat(3,1)=lat_min;
lat(4,1)=lat_min;
lat(5,1)=lat(1,1);
lon(1,1)=lon_min;
lon(2,1)=lon_max;
lon(3,1)=lon_max;
lon(4,1)=lon_min;
lon(5,1)=lon(1,1);
poligono{1,1}=get(handles.edit23,'stri
ng');
poligono{1,2}=get(handles.edit25,'stri
ng');
poligono{1,3}=lat;
poligono{1,4}=lon;
poligono{1,5}=lat;
poligono{1,6}=lon;
setappdata(handles.pushbutton1,'poligo
no',poligono)
set(handles.text37,'string',['N° Total
de pontos: ' num2str(numel(lat))])
set(handles.edit23,'Foregroundcolor','
b','string','Rectângulo')
set(handles.edit25,'Foregroundcolor','
b','string','Sigla')
poligono=getappdata(handles.pushbutton
1,'poligono');
area{1,1}=get(handles.checkbox16,'valu
e');
area{1,2}=poligono;
if get(handles.checkbox13,'value')==1
    area{1,3}=1;
elseif
get(handles.checkbox14,'value')==1
    area{1,3}=2;
else
    area{1,3}=3;
end
area{1,4}=str2double(get(handles.edit2
4,'string'));
setappdata(handles.pushbutton1,'area',
area)
% --- Executes on button press in
pushbutton9.
function pushbutton9_Callback(hObject,
eventdata, handles)
% hObject    handle to pushbutton9
(see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
%%%%%%%%%% Círculo %%%%%%%%%
idx=findobj('tag','regua');
if ~isempty(idx)
    delete(idx)
end
% Loop, picking up the points.
but = 1;
set(gcf,'CurrentAxes',handles.axes1)
%[xi,yi,but] =
inputm(1,handles.axes1);
%[xi, yi] = getpts(handles.axes1)
[xi,yi,but] = ginput(1);
pts=gcpmap;
lat = pts(1,1);lon=pts(1,2);
raio=60;
prompt={'Insira o raio do círculo
(milhas náuticas)'};
dlg_title='PESQUISA EM
CIRCUNFERÊNCIA';
```

```
num_lines=1;
defAns={'50'};
answer =
inputdlg(prompt,dlg_title,num_lines,de
fAns);
if ~isempty(answer)
    idx=findobj('tag','rect');
    if ~isempty(idx)
        delete(idx)
    end
    idx=findobj('tag','rmp');
    if ~isempty(idx)
        delete(idx)
    end
    idx=findobj('tag','pos');
    if ~isempty(idx)
        delete(idx)
    end
    idx=findobj('tag','pt');
    if ~isempty(idx)
        delete(idx)
    end
    raio=str2double(answer{1});
    az=0:180/12:360;
    for i=1:numel(az)
        [latout(i)
lonout(i)]=reckon(lat, lon,
nm2deg(raio), az(i));
    end
    plotm(latout,lonout,11,'--
r','tag','rect')
    poligono=[];
    latmean = lat;
    lonmean = lon;
h1=plotm(latmean,lonmean,'+k');set(h1,
'tag','pt');
    [ lat_str lon_str
]=str_deg(latmean,lonmean);
h=textm(latmean,lonmean,20,{lat_str;lo
n_str},
'FontSize',8);set(h,'tag','rect');
lat_max=max(latout);lat_min=min(latout
);
lon_max=max(lonout);lon_min=min(lonout
);
%criação do cellarray poligono
poligono{1,1}=get(handles.edit23,'stri
ng');
poligono{1,2}=get(handles.edit25,'stri
ng');
    poligono{1,3}=latout;
    poligono{1,4}=lonout;
    poligono{1,5}=latout;
    poligono{1,6}=lonout;
setappdata(handles.pushbutton1,'poligo
no',poligono)
set(handles.edit23,'Foregroundcolor','
b','string','Circulo')
set(handles.edit25,'Foregroundcolor','
b','string','Sigla')
set(handles.text37,'string',['N° Total
de Pontos: ' num2str(numel(latout))])
else
    idx=findobj('tag','rect');
    if ~isempty(idx)
        delete(idx)
    end
    idx=findobj('tag','rmp');
    if ~isempty(idx)
        delete(idx)
    end
    idx=findobj('tag','pos');
    if ~isempty(idx)
        delete(idx)
    end
    idx=findobj('tag','pt');
    if ~isempty(idx)
        delete(idx)
    end
    raio=50;
    az=0:180/12:360;
    for i=1:numel(az)
        [latout(i)
lonout(i)]=reckon(lat, lon,
nm2deg(raio), az(i));
    end
    plotm(latout,lonout,11,'--
r','tag','rect')
    poligono=[];
    latmean = lat;
    lonmean = lon;
h1=plotm(latmean,lonmean,'+k');set(h1,
'tag','pt');
    [ lat_str lon_str
]=str_deg(latmean,lonmean);
h=textm(latmean,lonmean,20,{lat_str;lo
n_str},
'FontSize',8);set(h,'tag','rect');
lat_max=max(latout);lat_min=min(latout
);
lon_max=max(lontout);lon_min=min(lonto
ut);
%criação do cellarray poligono
poligono{1,1}=get(handles.edit23,'stri
ng');
poligono{1,2}=get(handles.edit25,'stri
ng');
    poligono{1,3}=latout;
    poligono{1,4}=lonout;
    poligono{1,5}=latout;
    poligono{1,6}=lonout;
setappdata(handles.pushbutton1,'poligo
no',poligono)
set(handles.edit23,'Foregroundcolor','
b','string','Circulo')
set(handles.edit25,'Foregroundcolor','
b','string','Sigla')
set(handles.text37,'string',['N° Total
de pontos: ' num2str(numel(latout))])
end
poligono=getappdata(handles.pushbutton
1,'poligono');
area{1,1}=get(handles.checkbox16,'valu
e');
area{1,2}=poligono;
if get(handles.checkbox13,'value')==1
    area{1,3}=1;
elseif
get(handles.checkbox14,'value')==1
    area{1,3}=2;
else
    area{1,3}=3;
end
area{1,4}=str2double(get(handles.edit2
4,'string'));
```

```
setappdata(handles.pushbutton1,'area',
area)

% --- Executes on button press in
pushbutton10.
function
pushbutton10_Callback(hObject,
eventdata, handles)
% hObject    handle to pushbutton10
(see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Carregar ficheiros .MAT e .XLS %
dir_trabalho=getappdata(handles.pushbu
tton1,'dir_trabalho');
% Construct a questdlg with three
options
choice = questdlg('Pretende carregar
um polígono guardado em ficheiro
".mat" ou ".xls"?', ...
    'Menu de Carregamento', ...
    'MAT -
file','XLS','Cancelar','Cancelar');
% Handle response
dessert=0;
switch choice
    case 'MAT - file'
        disp([choice ' coming right
up.'])
        dessert = 1;
    case 'XLS'
        disp([choice ' coming right
up.'])
        dessert = 2;
    case 'Cancelar'
        disp('I''ll bring you your
check.')
        dessert = 0;
end
 if dessert ==1
    d = dir([dir_trabalho
'\poligonos\mat\*.mat']);
    str = {d.name};
[s,v]=listdlg('Name','Polígonos','Prom
ptstring','Polígonos gravados em
root/trabalho/poligonos',...
'SelectionMode','single','ListString',
str);
    if v==1
idx=findobj('tag','rect');
        if ~isempty(idx)
            delete(idx)
        end
        idx=findobj('tag','pt');
        if ~isempty(idx)
            delete(idx)
        end
        %try
            load([dir_trabalho
'\poligonos\mat\' str{s}])
setappdata(handles.pushbutton1,'poligo
no',poligono)
            if isempty(poligono)
            else
lat=poligono{1,5};
lon=poligono{1,6};
                if isempty(lat) ||
isempty(lon)
lat=poligono{1,3};
lon=poligono{1,4};
                end
latmean=mean(lat);lonmean=mean(lon);
h=plotm(lat,lon,35,'r--
','Linewidth',1.5);set(h,'tag','rect')
;
h1=plotm(latmean,lonmean,'+k');set(h1,
'tag','pt');
set(handles.edit23,'string',poligono{1
,1})
set(handles.edit25,'string',poligono{1
,2})
set(handles.text37,'string',['N° Total
de Pontos: '
num2str(numel(poligono{1,3}))])
            end
set(handles.edit23,'Foregroundcolor','
b')
set(handles.edit25,'Foregroundcolor','
b')
        %catch
        %end
    end
    idx=findobj('tag','rmp');
    if ~isempty(idx)
        delete(idx)
    end
    idx=findobj('tag','pos');
    if ~isempty(idx)
        delete(idx)
    end
elseif dessert==2
dir_trabalho=getappdata(handles.pushbu
tton1,'dir_trabalho');
    [filename pathname]=
uigetfile('*.xls','Directoria para a
pasta de polígonos em XLS');
    if isnumeric(filename) &&
filename==0
    else
        flag=0;
        try
[numeric,txt,raw]=xlsread([pathname
filename],'POLIGONO');
        catch
            flag=1;
        end
        if flag==1%Se não existe a
sheet POLIGONO
h = msgbox('O ficheiro não contém
informação válida!','ATENÇÃO!!!');
            waitfor(h)
        else
            try
% Leitura do polígono que está na
sheet "POLIGONO"
[n c]=size(raw);
for i=1:n
if strcmp(raw{i,1},'END')
  k=i;
      end
        end
latitude=cell2mat(raw(4:k-1,1));
longitude=cell2mat(raw(4:k-1,2));
%criação do cellarray poligono
```

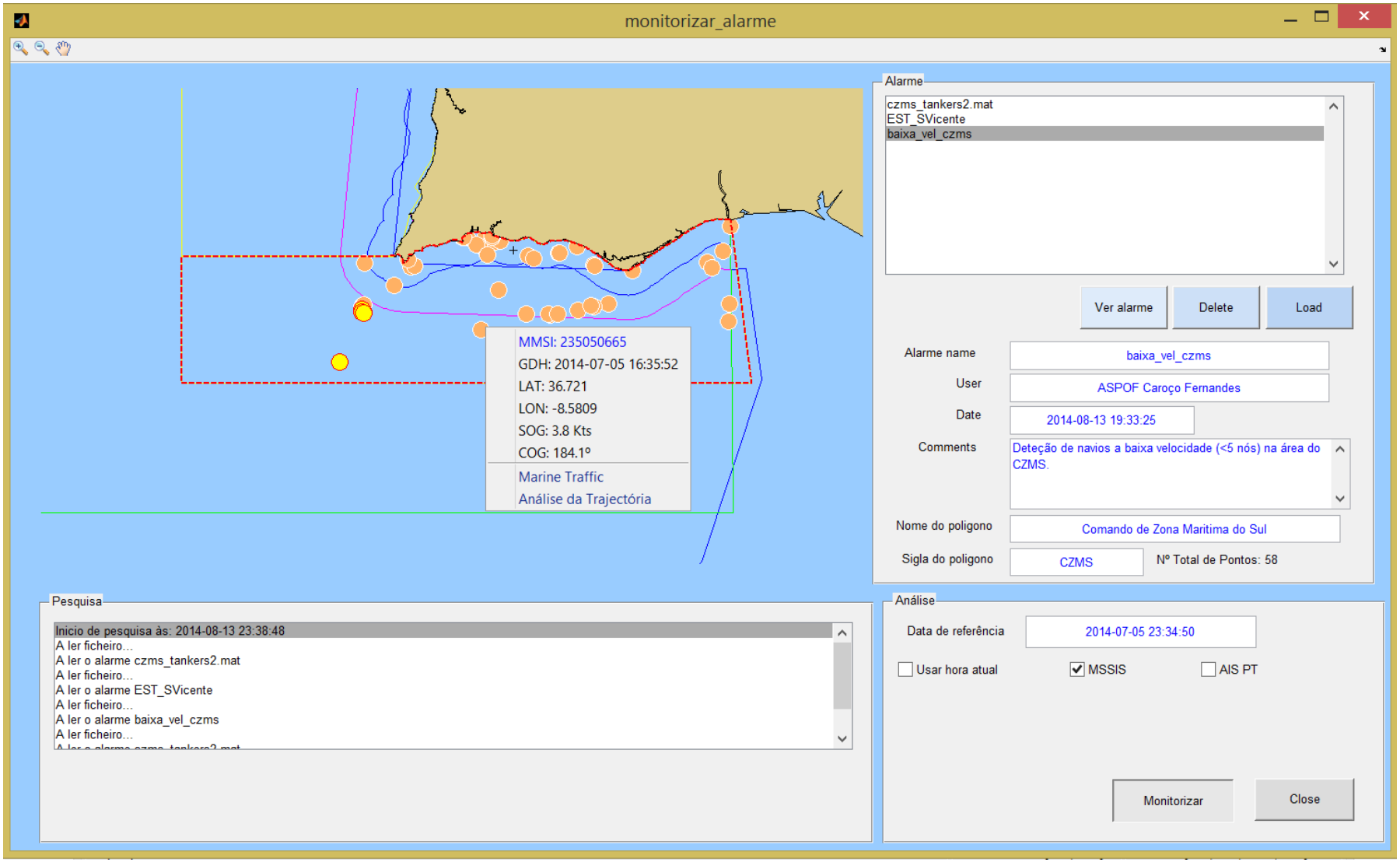
```
poligono{1,1}=raw{1,2};
poligono{1,2}=raw{2,2};
poligono{1,3}=latitude;
poligono{1,4}=longitude;
poligono{1,5}=latitude;
poligono{1,6}=longitude;
setappdata(handles.pushbutton1,'poligo
no',poligono)
 %DESNEHO DO POLIGONO
idx=findobj('tag','rect');
                if ~isempty(idx)
                    delete(idx)
                end
idx=findobj('tag','pt');
                if ~isempty(idx)
                    delete(idx)
                end
                %try
poligono=getappdata(handles.pushbutton
1,'poligono');
 if isempty(poligono)
else
lat=poligono{1,5};
lon=poligono{1,6};
  if isempty(lat) || isempty(lon)
lat=poligono{1,3};
lon=poligono{1,4};
                        end
latmean=mean(lat(1:end-
1));lonmean=mean(lon(1:end-1));
h=plotm(lat,lon,35,'r--
','Linewidth',1.5);set(h,'tag','rect')
;
h1=plotm(latmean,lonmean,'+k');set(h1,
'tag','pt');
set(handles.edit23,'string',poligono{1
,1})
set(handles.edit25,'string',poligono{1
,2})
set(handles.text37,'string',['N° Total
de pontos: '
num2str(numel(poligono{1,3}))])
                    end
set(handles.edit23,'Foregroundcolor','
b')
set(handles.edit25,'Foregroundcolor','
b')
idx=findobj('tag','rmp');
                if ~isempty(idx)
                    delete(idx)
                end
idx=findobj('tag','pos');
                if ~isempty(idx)
                    delete(idx)
                end
            catch
 h = msgbox('O ficheiro não contém
informação válida!','ATENÇÃO!!!');
                waitfor(h)
            end
        end
    end
end
poligono=getappdata(handles.pushbutton
1,'poligono');
area{1,1}=get(handles.checkbox16,'valu
e');
area{1,2}=poligono;
if get(handles.checkbox13,'value')==1
    area{1,3}=1;
elseif
get(handles.checkbox14,'value')==1
    area{1,3}=2;
else
    area{1,3}=3;
end
area{1,4}=str2double(get(handles.edit2
4,'string'));
setappdata(handles.pushbutton1,'area',
area)
% --- Executes on button press in
checkbox17.
function checkbox17_Callback(hObject,
eventdata, handles)
% hObject    handle to checkbox17 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns
toggle state of checkbox17
if get(handles.checkbox17,'value')==1
set(handles.edit26,'enable','on')
set(handles.edit27,'enable','on')
set(handles.edit28,'enable','on')
set(handles.edit29,'enable','on')
set(handles.pushbutton12,'enable','on'
)
else
set(handles.checkbox17,'value',0)
set(handles.edit26,'enable','off')
set(handles.edit27,'enable','off')
set(handles.edit28,'enable','off')
set(handles.edit29,'enable','off')
set(handles.pushbutton12,'enable','off
')
end
gdh{1,1}=get(handles.checkbox17,'value
');
gdh{1,2}{1,1}='validade';
gdh{1,2}{1,2}=datenum(get(handles.edit
26,'string'),'yyyy-mm-dd HH:MM:SS');
gdh{1,2}{1,3}=datenum(get(handles.edit
27,'string'),'yyyy-mm-dd HH:MM:SS');
gdh{1,2}{2,1}='periodo horário';
gdh{1,2}{2,2}=get(handles.edit28,'stri
ng');
gdh{1,2}{2,3}=get(handles.edit29,'stri
ng');
setappdata(handles.pushbutton1,'gdh',g
dh)
gdh
function edit26_Callback(hObject,
eventdata, handles)
% hObject  handle to edit26 (see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hints: get(hObject,'String') returns
contents of edit26 as text
str2double(get(hObject,'String'))
returns contents of edit26 as a double
set(handles.edit26,'Foregroundcolor','
b')
```

```
set(handles.edit27,'Foregroundcolor','
b')
set(handles.edit28,'Foregroundcolor','
b')
set(handles.edit29,'Foregroundcolor','
b')

% --- Executes during object creation,
after setting all properties.
function edit26_CreateFcn(hObject,
eventdata, handles)
% hObject    handle to edit26 (see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    empty - handles not
created until after all CreateFcns
called
% Hint: edit controls usually have a
white background on Windows.
%       See ISPC and COMPUTER.
if ispc &&
isequal(get(hObject,'BackgroundColor')
,
get(0,'defaultUicontrolBackgroundColor
'))
set(hObject,'BackgroundColor','white')
;
end
function edit27_Callback(hObject,
eventdata, handles)
% hObject    handle to edit27 (see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hints: get(hObject,'String') returns
contents of edit27 as text
str2double(get(hObject,'String'))
returns contents of edit27 as a double
set(handles.edit26,'Foregroundcolor','
b')
set(handles.edit27,'Foregroundcolor','
b')
set(handles.edit28,'Foregroundcolor','
b')
set(handles.edit29,'Foregroundcolor','
b')
d1=get(handles.edit26,'string')
d1=datenum(d1,'yyyy-mm-dd HH:MM:SS');
d2=get(handles.edit27,'string')
d2=datenum(d2,'yyyy-mm-dd HH:MM:SS');
dias=round(d2-d1+1);
set(handles.text33,'string',['Total em
dias: ' num2str(dias)])

% --- Executes during object creation,
after setting all properties.
function edit27_CreateFcn(hObject,
eventdata, handles)
% hObject    handle to edit27 (see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    empty - handles not
created until after all CreateFcns
called
% Hint: edit controls usually have a
white background on Windows.
%       See ISPC and COMPUTER.
if ispc &&
isequal(get(hObject,'BackgroundColor')
,
get(0,'defaultUicontrolBackgroundColor
'))
set(hObject,'BackgroundColor','white')
;
end
function edit28_Callback(hObject,
eventdata, handles)
% hObject    handle to edit28 (see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hints: get(hObject,'String') returns
contents of edit28 as text
str2double(get(hObject,'String'))
returns contents of edit28 as a double
set(handles.edit26,'Foregroundcolor','
b')
set(handles.edit27,'Foregroundcolor','
b')
set(handles.edit28,'Foregroundcolor','
b')
set(handles.edit29,'Foregroundcolor','
b')
% --- Executes during object creation,
after setting all properties.
function edit28_CreateFcn(hObject,
eventdata, handles)
% hObject    handle to edit28 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    empty - handles not
created until after all CreateFcns
called
% Hint: edit controls usually have a
white background on Windows.
%       See ISPC and COMPUTER.
if ispc &&
isequal(get(hObject,'BackgroundColor')
,
get(0,'defaultUicontrolBackgroundColor
'))
set(hObject,'BackgroundColor','white')
;
end
function edit29_Callback(hObject,
eventdata, handles)
% hObject    handle to edit29 (see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hints: get(hObject,'String') returns
contents of edit29 as text
str2double(get(hObject,'String'))
returns contents of edit29 as a double
set(handles.edit26,'Foregroundcolor','
b')
set(handles.edit27,'Foregroundcolor','
b')
set(handles.edit28,'Foregroundcolor','
b')
set(handles.edit29,'Foregroundcolor','
b')
```

```
% --- Executes during object creation,
after setting all properties.
function edit29_CreateFcn(hObject,
eventdata, handles)
% hObject  handle to edit29 (see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    empty - handles not
created until after all CreateFcns
called
% Hint: edit controls usually have a
white background on Windows.
%       See ISPC and COMPUTER.
if ispc &&
isequal(get(hObject,'BackgroundColor')
,
get(0,'defaultUicontrolBackgroundColor
'))
set(hObject,'BackgroundColor','white')
;
end
% --- Executes on button press in
pushbutton11.
function
pushbutton11_Callback(hObject,
eventdata, handles)
% hObject    handle to pushbutton11
(see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
if
isnan(str2double(get(handles.edit1,'st
ring')))
    h = warndlg('Insira um MMSI
válido!');
    waitfor(h)
else
% Construct a questdlg with three
options
    choice = questdlg('Pretende aceder
à BD do AISINTEL ou ao website Marine
Traffic?', ...
        'Menu de Visualização',
        'BD AISINTEL','Marine
Traffic','Cancelar','Cancelar');
    % Handle response
    dessert=0;
    switch choice
        case 'BD AISINTEL'
            disp([choice ' coming
right up.'])
            dessert = 1;
        case 'Marine Traffic'
            disp([choice ' coming
right up.'])
            dessert = 2;
        case 'No thank you'
  disp('I''ll bring you your check.')
            dessert = 0;
    end
    if dessert==1
ver_navio(get(handles.edit1,'string'))
    elseif dessert==2
mmsi=str2double(get(handles.edit1,'str
ing'));
%winopen(['C:\Documents and
Settings\user\Ambiente de
trabalho\teste.html'])
dir_trabalho=getappdata(handles.pushbu
tton1,'dir_trabalho');
        a{1}='<html>';
        a{2}='<head>';
        a{3}=['<meta http-
equiv="refresh"
content="0;url=http://www.marinetraffi
c.com/ais/pt/shipdetails.aspx?mmsi='
num2str(mmsi) '" />'];
        a{4}='</head>';
        a{5}='<body>teste';
        a{6}='</body>';
        a{7}='</html>';
        fid=fopen([dir_trabalho
'teste.html'],'wt');
        for i=1:7
            fprintf(fid, '%s\n',a{i});
        end
        fclose(fid);
        winopen([dir_trabalho
'teste.html'])
        %}
    else
    end
end
% --- Executes when selected cell(s)
is changed in uitable1.
function
uitable1_CellSelectionCallback(hObject
, eventdata, handles)
% hObject handle to uitable1 (see
GCBO)
% eventdata  structure with the
following fields (see UITABLE)
%   Indices: row and column indices of
the cell(s) currently selecteds
% handles    structure with handles
and user data (see GUIDATA)
lista_navios=getappdata(handles.pushbu
tton1,'lista_navios');
if ~isempty(eventdata.Indices) &&
~isempty(lista_navios)
    [n m]=size(lista_navios);
    l=eventdata.Indices(1);%indice de
linha individuo
    if l>n
        l=n;
    end
set(handles.uitable1,'userdata',l)
set(handles.edit1,'string',lista_navio
s{l,1})
set(handles.checkbox4,'value',lista_na
vios{l,2})
set(handles.edit3,'string',lista_navio
s{l,3})
set(handles.edit4,'string',lista_navio
s{l,4})
set(handles.checkbox5,'value',lista_na
vios{l,5})
set(handles.edit5,'string',lista_navio
s{l,6})
set(handles.edit6,'string',lista_navio
s{l,7})
end
```

```
% --- Executes during object creation,
after setting all properties.
function text33_CreateFcn(hObject,
eventdata, handles)
% hObject    handle to text33 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    empty - handles not
created until after all CreateFcns
called
% --- Executes on button press in
pushbutton12.
function
pushbutton12_Callback(hObject,
eventdata, handles)
% hObject    handle to pushbutton12
(see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
set(handles.edit26,'Foregroundcolor','
b')
set(handles.edit27,'Foregroundcolor','
b')
set(handles.edit28,'Foregroundcolor','
b')
set(handles.edit29,'Foregroundcolor','
b')
gdh{1,1}=get(handles.checkbox17,'value
');
gdh{1,2}{1,1}='validade';
gdh{1,2}{1,2}=datenum(get(handles.edit
26,'string'),'yyyy-mm-dd HH:MM:SS');
gdh{1,2}{1,3}=datenum(get(handles.edit
27,'string'),'yyyy-mm-dd HH:MM:SS');
gdh{1,2}{2,1}='periodo horário';
gdh{1,2}{2,2}=get(handles.edit28,'stri
ng');
gdh{1,2}{2,3}=get(handles.edit29,'stri
ng');
setappdata(handles.pushbutton1,'gdh',g
dh)
% --- Executes during object creation,
after setting all properties.
function checkbox16_CreateFcn(hObject,
eventdata, handles)
% hObject    handle to checkbox16 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    empty - handles not
created until after all CreateFcns
called
% --- Executes when entered data in
editable cell(s) in uitable1.
function
uitable1_CellEditCallback(hObject,
eventdata, handles)
% hObject    handle to uitable1 (see
GCBO)
% eventdata  structure with the
following fields (see UITABLE)
%   Indices: row and column indices of
the cell(s) edited
%   PreviousData: previous data for
the cell(s) edited
%   EditData: string(s) entered by the
user
%   NewData: EditData or its converted
form set on the Data property. Empty
if Data was not changed
%   Error: error string when failed to
convert EditData to appropriate value
for Data
% handles    structure with handles
and user data (see GUIDATA)
% --- Executes on button press in
pushbutton13.
function
pushbutton13_Callback(hObject,
eventdata, handles)
% hObject    handle to pushbutton13
(see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
monitorizar_alarme
% --- Executes on selection change in
popupmenu2.
function popupmenu2_Callback(hObject,
eventdata, handles)
% hObject    handle to popupmenu2 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hints: contents =
cellstr(get(hObject,'String')) returns
popupmenu2 contents as cell array
contents{get(hObject,'Value')} returns
selected item from popupmenu2
set(handles.popupmenu2,'Foregroundcolo
r','b')
% --- Executes during object creation,
after setting all properties.
function popupmenu2_CreateFcn(hObject,
eventdata, handles)
% hObject    handle to popupmenu2 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    empty - handles not
created until after all CreateFcns
called
% Hint: popupmenu controls usually
have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc &&
isequal(get(hObject,'BackgroundColor')
,
get(0,'defaultUicontrolBackgroundColor
'))
set(hObject,'BackgroundColor','white')
;
end
```

Anexo C – Interface de monitorização



Anexo D – Script da interface de monitorização

```
function varargout =
monitorizar_alarme(varargin)
% MONITORIZAR_ALARME MATLAB code for
monitorizar_alarme.fig
%      MONITORIZAR_ALARME, by itself,
creates a new MONITORIZAR_ALARME or
raises the existing
%      singleton*.
%      H = MONITORIZAR_ALARME returns
the handle to a new MONITORIZAR_ALARME
or the handle to
%      the existing singleton*.
MONITORIZAR_ALARME('CALLBACK',hObject,
eventData,handles,...) calls the local
%      function named CALLBACK in
MONITORIZAR_ALARME.M with the given
input arguments.
MONITORIZAR_ALARME('Property','Value',
...) creates a new MONITORIZAR_ALARME
or raises the
%      existing singleton*.  Starting
from the left, property value pairs
are
%      applied to the GUI before
monitorizar_alarme_OpeningFcn gets
called.  An
%      unrecognized property name or
invalid value makes property
application
%      stop.  All inputs are passed to
monitorizar_alarme_OpeningFcn via
varargin.
%      *See GUI Options on GUIDE's
Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the
response to help monitorizar_alarme
% Last Modified by GUIDE v2.5 07-Apr-
2014 12:10:29
% Begin initialization code - DO NOT
EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',
mfilename, ...
  'gui_Singleton',  gui_Singleton, ...
 'gui_OpeningFcn',
@monitorizar_alarme_OpeningFcn, ...
                   'gui_OutputFcn',
@monitorizar_alarme_OutputFcn, ...
  'gui_LayoutFcn',  [] , ...
              'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback =
str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] =
gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State,
varargin{:});
end
% End initialization code - DO NOT
EDIT

% --- Executes just before
monitorizar_alarme is made visible.
function
monitorizar_alarme_OpeningFcn(hObject,
eventdata, handles, varargin)
% This function has no output args,
see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% varargin   command line arguments to
monitorizar_alarme (see VARARGIN)
% Choose default command line output
for monitorizar_alarme
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
%%%%%% MAPAS %%%%%%%
handles.axes1_latlim = [-16.5 60];
handles.axes1_lonlim = [-75 50];
handles.axes1_latlimp = [-16.5 60];
handles.axes1_lonlimp = [-75 50];
%directoria para a pasta de dados
if
exist('caminho_dados_ais.mat','file')
load caminho_dados_ais%carrega
dir_dados
setappdata(handles.pushbutton1,'dir_da
dos',dir_dados);%'C:\Documents and
Settings\user\Os meus
documentos\MATLAB\trabalho
else
    dir_dados =
uigetdir(pwd,'Directoria para a pasta
de dados (BDAIS)');
setappdata(handles.pushbutton1,'dir_da
dos',dir_dados);
    save caminho_dados_ais dir_dados
end
%directoria para a pasta MONICAP
if
exist('caminho_dados_monicap.mat','fil
e')
    load caminho_dados_monicap%carrega
dir_trabalho
setappdata(handles.pushbutton1,'dir_mo
nicap',dir_monicap);%'C:\Documents and
```

```
Settings\user\Os meus
documentos\MATLAB\trabalho
else
    dir_monicap =
uigetdir(pwd,'Directoria para a pasta
de dados MONICAP');
setappdata(handles.pushbutton1,'dir_mo
nicap',dir_monicap);
    save caminho_dados_monicap
dir_monicap
end
%directoria para a pasta FOTOS
if exist('caminho_fotos.mat','file')
    load caminho_fotos%carrega
dir_fotos
setappdata(handles.pushbutton1,'dir_fo
tos',dir_fotos);%'C:\Documents and
Settings\user\Os meus
documentos\MATLAB\trabalho
else
    dir_fotos =
uigetdir(pwd,'Directoria para a pasta
de FOTOS');
setappdata(handles.pushbutton1,'dir_fo
tos',dir_fotos);
    save caminho_fotos dir_fotos
end
%directoria para a pasta de trabalho
if
exist('caminho_trabalho_ais.mat','file
')
    load caminho_trabalho_ais%carrega
dir_trabalho
setappdata(handles.pushbutton1,'dir_tr
abalho',dir_trabalho);%'C:\Documents
and Settings\user\Os meus
documentos\MATLAB\trabalho
else
    dir_trabalho =
uigetdir(pwd,'Directoria para a pasta
de trabalho');
setappdata(handles.pushbutton1,'dir_tr
abalho',dir_trabalho);
    save caminho_trabalho_ais
dir_trabalho
end
%axes(handles.axes1)
set(gcf, 'Renderer', 'painters');
%ax=worldmap(handles.axes1_latlim,hand
les.axes1_lonlim);set(ax,'tag','mapa')
;
ax=worldmap('world');
setm(ax,'mapprojection','eqdcylin')
%setm(ax,'FFaceColor',([0.941 0.941
0.941]),'Grid','on')
setm(ax,'FFaceColor',([0.6 0.8
1]),'Grid','on')
load ([dir_trabalho
'\poligonos_reduced_c200']);%
warning off
h=patchesm1(latreduced2,lonreduced2,10
,[222/255 205/255 139/255]);
%load ('C:\Documents and
Settings\user\Os meus
documentos\MATLAB\trabalho\poligono_ma
pa');
%h=patchesm(poligono_mapa(:,1),poligon
o_mapa(:,2),10,[222/255 205/255
139/255]);
%load ('C:\Documents and
Settings\user\Os meus
documentos\MATLAB\trabalho\poligono_ma
pas_reduced');
%h=patchesm(lat,lon,10,[222/255
205/255 139/255]);
set(h,'Tag','coast');
load([dir_trabalho
'\tabela_areas.mat'])
load([dir_trabalho '\tabela_mes.mat'])
h2=plotm(tabela_areas{2,3},tabela_area
s{2,4},-2);
set(h2,'color','k');
h3=plotm(tabela_areas{1,3},tabela_area
s{1,4},-2);
set(h3,'color','b');
h4=plotm(tabela_areas{3,3},tabela_area
s{3,4},-2);
set(h4,'color','m');
h5=plotm(tabela_areas{6,3},tabela_area
s{6,4},-2);
set(h5,'color','y');
h6=plotm(tabela_areas{7,3},tabela_area
s{7,4},-2);
set(h6,'color','y');
h7=plotm(tabela_areas{8,3},tabela_area
s{8,4},-2);
set(h7,'color','y');
h8=plotm(tabela_areas{9,3},tabela_area
s{9,4},-2);
set(h8,'color','g');
h9=plotm(tabela_areas{10,3},tabela_are
as{10,4},-2);
set(h9,'color','g');
%h10=linem([45;-40],[29;-40]);
%set(h10,'color','g');
setm(ax,'fontsize',6)
tightmap
%%%%%%%%%%%%%%%%FIM MAPAS
d=now;
set(handles.edit2,'string',datestr(d,3
1),'Foregroundcolor','b')
alarme_global=[]
setappdata(handles.pushbutton1,'alarme
_global',alarme_global)
L={'NIL'};
set(handles.listbox1,'string',L);
set(handles.edit2,'enable','off')
% UIWAIT makes monitorizar_alarme wait
for user response (see UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are
returned to the command line.
function varargout =
monitorizar_alarme_OutputFcn(hObject,
eventdata, handles)
% varargout  cell array for returning
output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Get default command line output from
handles structure
```

```
varargout{1} = handles.output;
function edit2_Callback(hObject,
eventdata, handles)
% hObject    handle to edit2 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hints: get(hObject,'String') returns
contents of edit2 as text
str2double(get(hObject,'String'))
returns contents of edit2 as a double

% --- Executes during object creation,
after setting all properties.
function edit2_CreateFcn(hObject,
eventdata, handles)
% hObject  handle to edit2 (see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    empty - handles not
created until after all CreateFcns
called
% Hint: edit controls usually have a
white background on Windows.
%       See ISPC and COMPUTER.
if ispc &&
isequal(get(hObject,'BackgroundColor')
,
get(0,'defaultUicontrolBackgroundColor
'))
set(hObject,'BackgroundColor','white')
;
end

% --- Executes on button press in
checkbox1.
function checkbox1_Callback(hObject,
eventdata, handles)
% hObject    handle to checkbox1 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns
toggle state of checkbox1
if get(handles.checkbox1,'value')==1
set(handles.edit2,'enable','off')
else
    set(handles.edit2,'enable','on')
end
function edit1_Callback(hObject,
eventdata, handles)
% hObject    handle to edit1 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hints: get(hObject,'String') returns
contents of edit1 as text
str2double(get(hObject,'String'))
returns contents of edit1 as a double

% --- Executes during object creation,
after setting all properties.
function edit1_CreateFcn(hObject,
eventdata, handles)
% hObject  handle to edit1 (see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    empty - handles not
created until after all CreateFcns
called
% Hint: edit controls usually have a
white background on Windows.
%       See ISPC and COMPUTER.
if ispc &&
isequal(get(hObject,'BackgroundColor')
,
get(0,'defaultUicontrolBackgroundColor
'))
set(hObject,'BackgroundColor','white')
;
end

% --- Executes on selection change in
listbox1.
function listbox1_Callback(hObject,
eventdata, handles)
% hObject    handle to listbox1 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hints: contents =
cellstr(get(hObject,'String')) returns
listbox1 contents as cell array
contents{get(hObject,'Value')} returns
selected item from listbox1

% --- Executes during object creation,
after setting all properties.
function listbox1_CreateFcn(hObject,
eventdata, handles)
% hObject handle to listbox1 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    empty - handles not
created until after all CreateFcns
called
% Hint: listbox controls usually have
a white background on Windows.
%       See ISPC and COMPUTER.
if ispc &&
isequal(get(hObject,'BackgroundColor')
,
get(0,'defaultUicontrolBackgroundColor
'))
set(hObject,'BackgroundColor','white')
;
end

% --- Executes on selection change in
listbox2.
function listbox2_Callback(hObject,
eventdata, handles)
% hObject    handle to listbox2 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
```

```
% handles    structure with handles
and user data (see GUIDATA)
% Hints: contents =
cellstr(get(hObject,'String')) returns
listbox2 contents as cell array
contents{get(hObject,'Value')} returns
selected item from listbox2
idx=get(handles.listbox2,'value')
alarme_global=getappdata(handles.pushb
utton1, 'alarme_global')
if  ~isempty(alarme_global)
    alarme=alarme_global{idx,2}
    %carregar número de pontos do
poligono----------------
    poligono=alarme.area{1,2};
    if ~isempty(poligono)
        idx=findobj('tag','rect');
        if ~isempty(idx)
            delete(idx)
        end
        idx=findobj('tag','pt');
        if ~isempty(idx)
            delete(idx)
        end
        poligono
set(handles.text11,'string',['N° Total
de Pontos: '
num2str(numel(poligono{1,3}))])
        lat=poligono{1,5};
        lon=poligono{1,6};
        if isempty(lat) ||
isempty(lon)
            lat=poligono{1,3};
            lon=poligono{1,4};
        end
%carregar poligonos no mapa------
latmean=mean(lat);lonmean=mean(lon);
  h=plotm(lat,lon,35,'r--
','Linewidth',1.5);set(h,'tag','rect')
;
h1=plotm(latmean,lonmean,'+k');set(h1,
'tag','pt');
    else
set(handles.text11,'string',['N° Total
de Pontos: ' num2str(0)])
    end
set(handles.edit1,'string',alarme.file
name,'Foregroundcolor','b')
set(handles.edit9,'string',alarme.user
,'Foregroundcolor','b')
set(handles.edit8,'string',datestr(ala
rme.data,'yyyy-mm-dd
HH:MM:SS'),'Foregroundcolor','b')
set(handles.edit10,'string',alarme.com
ments,'Foregroundcolor','b')
set(handles.edit11,'string',alarme.are
a{1,2}{1,1},'Foregroundcolor','b')
set(handles.edit12,'string',alarme.are
a{1,2}{1,2},'Foregroundcolor','b')
%set(handles.text11,'string',alarme.fi
lename,'Foregroundcolor','b')
end

% --- Executes during object creation,
after setting all properties.
function listbox2_CreateFcn(hObject,
eventdata, handles)
% hObject    handle to listbox2 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    empty - handles not
created until after all CreateFcns
called
% Hint: listbox controls usually have
a white background on Windows.
%       See ISPC and COMPUTER.
if ispc &&
isequal(get(hObject,'BackgroundColor')
,
get(0,'defaultUicontrolBackgroundColor
'))
set(hObject,'BackgroundColor','white')
;
end

% --- Executes on button press in
pushbutton1.
function pushbutton1_Callback(hObject,
eventdata, handles)
% hObject    handle to pushbutton1
(see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
%%%%%%% Load %%%%%%%%
dir_trabalho=getappdata(handles.pushbu
tton1,'dir_trabalho');
[filename pathname] =
uigetfile({[dir_trabalho
'\alarmes\*.mat']},'');
if isnumeric(filename) && filename==0
else
    %Ler ficheiro EXCEL
    load([pathname filename])%carrega
qeo e cabecalho
    who
    alarme
    save treta1 alarme
%carregar nome do alarme---
    if ~isempty(alarme.filename)
        alarme.filename
set(handles.edit1,'string',alarme.file
name,'Foregroundcolor','b')
    else
set(handles.edit1,'string','Dados não
disponiveis','Foregroundcolor','b')
    end
%carregar data do alarme-----
    if ~isempty(alarme.data)
        alarme.data
set(handles.edit8,'string',datestr(ala
rme.data,'yyyy-mm-dd
HH:MM:SS'),'Foregroundcolor','b')
    else
set(handles.edit8,'string',['Data não
disponivel'],'Foregroundcolor','b')
    end
%carregar nome utilizador que criou o
alarme -------------
    if ~isempty(alarme.user)
set(handles.edit9,'string',alarme.user
,'Foregroundcolor','b')
    else
```

```
set(handles.edit9,'string',['User não
disponivel'],'Foregroundcolor','b')
    end
%carregar comentários do alarme-----
    if ~isempty(alarme.comments)
set(handles.edit10,'string',alarme.com
ments,'Foregroundcolor','b')
    else
set(handles.edit10,'string',['Comentár
ios não
disponiveis'],'Foregroundcolor','b')
    end
%carregar nome e sigla do poligono--
    if ~isempty(alarme.area{1,2})
%set(handles.text11,'string',['N° de
pontos disponíveis: '
num2str(numel(alarme.area{1,2}{1,4}))]
,'Foregroundcolor','b')
set(handles.edit11,'string',alarme.are
a{1,2}{1,1},'Foregroundcolor','b')
set(handles.edit12,'string',alarme.are
a{1,2}{1,2},'Foregroundcolor','b')
    else
set(handles.edit11,'string','poligono'
,'Foregroundcolor','b')
set(handles.edit12,'string','sigla','F
oregroundcolor','b')
    end
%carregar número de pontos do
poligono---
    poligono=alarme.area{1,2};
    if ~isempty(poligono)
%limpar os poligonos do mapa---
idx=findobj('tag','rect');
            if ~isempty(idx)
            delete(idx)
            end
            idx=findobj('tag','pt');
            if ~isempty(idx)
                delete(idx)
            end
        poligono
set(handles.text11,'string',['N° Total
de Pontos: '
num2str(numel(poligono{1,3}))])
        lat=poligono{1,5};
        lon=poligono{1,6};
        if isempty(lat) ||
isempty(lon)
            lat=poligono{1,3};
            lon=poligono{1,4};
        end
%carregar poligonos no mapa---
latmean=mean(lat);lonmean=mean(lon);
        h=plotm(lat,lon,35,'r--
','Linewidth',1.5);set(h,'tag','rect')
;
h1=plotm(latmean,lonmean,'+k');set(h1,
'tag','pt');
    else
set(handles.text11,'string',['N° Total
de Pontos: ' num2str(0)])
    end
alarme_global=getappdata(handles.pushb
utton1,'alarme_global')
    [n m]=size(alarme_global)
    if n==0
alarme_global{1,1}=alarme.filename
        alarme_global{1,2}=alarme
    else
alarme_global{n+1,1}=alarme.filename
        alarme_global{n+1,2}=alarme
    end
setappdata(handles.pushbutton1,'alarme
_global',alarme_global)
set(handles.listbox2,'string',alarme_g
lobal(:,1))
end
function edit8_Callback(hObject,
eventdata, handles)
% hObject handle to edit8 (see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hints: get(hObject,'String') returns
contents of edit8 as text
str2double(get(hObject,'String'))
returns contents of edit8 as a double

% --- Executes during object creation,
after setting all properties.
function edit8_CreateFcn(hObject,
eventdata, handles)
% hObject handle to edit8 (see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    empty - handles not
created until after all CreateFcns
called
% Hint: edit controls usually have a
white background on Windows.
%       See ISPC and COMPUTER.
if ispc &&
isequal(get(hObject,'BackgroundColor')
,
get(0,'defaultUicontrolBackgroundColor
'))
set(hObject,'BackgroundColor','white')
;
end
function edit9_Callback(hObject,
eventdata, handles)
% hObject  handle to edit9 (see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hints: get(hObject,'String') returns
contents of edit9 as text
str2double(get(hObject,'String'))
returns contents of edit9 as a double

% --- Executes during object creation,
after setting all properties.
function edit9_CreateFcn(hObject,
eventdata, handles)
% hObject  handle to edit9 (see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    empty - handles not
created until after all CreateFcns
called
% Hint: edit controls usually have a
white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc &&
isequal(get(hObject,'BackgroundColor')
,
get(0,'defaultUicontrolBackgroundColor
'))
set(hObject,'BackgroundColor','white')
;
end
function edit10_Callback(hObject,
eventdata, handles)
% hObject handle to edit10 (see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hints: get(hObject,'String') returns
contents of edit10 as text
str2double(get(hObject,'String'))
returns contents of edit10 as a double

% --- Executes during object creation,
after setting all properties.
function edit10_CreateFcn(hObject,
eventdata, handles)
% hObject  handle to edit10 (see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    empty - handles not
created until after all CreateFcns
called
% Hint: edit controls usually have a
white background on Windows.
%       See ISPC and COMPUTER.
if ispc &&
isequal(get(hObject,'BackgroundColor')
,
get(0,'defaultUicontrolBackgroundColor
'))
set(hObject,'BackgroundColor','white')
;
end

% --- Executes on button press in
pushbutton2.
function pushbutton2_Callback(hObject,
eventdata, handles)
% hObject    handle to pushbutton2
(see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
%%%%%Delete %%%%%
idx=get(handles.listbox2,'value')
alarme_global=getappdata(handles.pushb
utton1, 'alarme_global')
if  ~isempty(alarme_global)
    alarme_global(idx,:)=[];
    [ns ms]=size(alarme_global);
    if  ns>0
        alarme=alarme_global{1,2};
     %carregar nome do alarme---
        if ~isempty(alarme.filename)
            alarme.filename
set(handles.edit1,'string',alarme.file
name,'Foregroundcolor','b')
        else
set(handles.edit1,'string','Dados não
disponiveis','Foregroundcolor','b')
        end
%carregar data do alarme---
        if ~isempty(alarme.data)
set(handles.edit8,'string',datestr(ala
rme.data,'yyyy-mm-dd
HH:MM:SS'),'Foregroundcolor','b')
        else
set(handles.edit8,'string',['Data não
disponivel'],'Foregroundcolor','b')
        end
%carregar nome utilizador que criou o
alarme --
        if ~isempty(alarme.user)
set(handles.edit9,'string',alarme.user
,'Foregroundcolor','b')
        else
set(handles.edit9,'string',['User não
disponivel'],'Foregroundcolor','b')
        end
%carregar comentários do alarme---
        if ~isempty(alarme.comments)
set(handles.edit10,'string',alarme.com
ments,'Foregroundcolor','b')
        else
set(handles.edit10,'string',['Comentár
ios não
disponiveis'],'Foregroundcolor','b')
        end
%carregar nome e sigla do poligono---
        if ~isempty(alarme.area{1,2})
set(handles.edit11,'string',alarme.are
a{1,2}{1,1},'Foregroundcolor','b')
set(handles.edit12,'string',alarme.are
a{1,2}{1,2},'Foregroundcolor','b')
        else
set(handles.edit11,'string','poligono'
,'Foregroundcolor','b')
set(handles.edit12,'string','sigla','F
oregroundcolor','b')
        end
        poligono=alarme.area{1,2};
        if ~isempty(poligono)
            idx=findobj('tag','rect');
            if ~isempty(idx)
                delete(idx)
            end
            idx=findobj('tag','pt');
            if ~isempty(idx)
                delete(idx)
            end
set(handles.text11,'string',['N° Total
de Pontos: '
num2str(numel(poligono{1,3}))])
            lat=poligono{1,5};
            lon=poligono{1,6};
            if isempty(lat) ||
isempty(lon)
                lat=poligono{1,3};
                lon=poligono{1,4};
            end
latmean=mean(lat);lonmean=mean(lon);
 h=plotm(lat,lon,35,'r--
','Linewidth',1.5);set(h,'tag','rect')
;
h1=plotm(latmean,lonmean,'+k');set(h1,
'tag','pt');
```

```
        else
set(handles.text11,'string',['N° Total
de Pontos: ' num2str(0)])
        end
    else
        idx=findobj('tag','rect');
        if ~isempty(idx)
            delete(idx)
        end
        idx=findobj('tag','pt');
        if ~isempty(idx)
            delete(idx)
        end
set(handles.text11,'string','')
set(handles.edit11,'string','','Foregr
oundcolor','b')
set(handles.edit12,'string','','Foregr
oundcolor','b')
set(handles.edit10,'string','','Foregr
oundcolor','b')
set(handles.edit1,'string','','Foregro
undcolor','b')
set(handles.edit9,'string','','Foregro
undcolor','b')
set(handles.edit8,'string','','Foregro
undcolor','b')
    end
setappdata(handles.pushbutton1,'alarme
_global',alarme_global)
set(handles.listbox2,'string',alarme_g
lobal(:,1),'value',1)
else
    idx=findobj('tag','rect');
    if ~isempty(idx)
        delete(idx)
    end
    idx=findobj('tag','pt');
    if ~isempty(idx)
        delete(idx)
    end
    xx=msgbox('Não esxitem mais
alarmes para apagar')
end
function edit11_Callback(hObject,
eventdata, handles)
% hObject  handle to edit11 (see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hints: get(hObject,'String') returns
contents of edit11 as text
str2double(get(hObject,'String'))
returns contents of edit11 as a double

% --- Executes during object creation,
after setting all properties.
function edit11_CreateFcn(hObject,
eventdata, handles)
% hObject  handle to edit11 (see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    empty - handles not
created until after all CreateFcns
called
% Hint: edit controls usually have a
white background on Windows.
%       See ISPC and COMPUTER.
if ispc &&
isequal(get(hObject,'BackgroundColor')
,
get(0,'defaultUicontrolBackgroundColor
'))
set(hObject,'BackgroundColor','white')
;
end
function edit12_Callback(hObject,
eventdata, handles)
% hObject  handle to edit12 (see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hints: get(hObject,'String') returns
contents of edit12 as text
str2double(get(hObject,'String'))
returns contents of edit12 as a double

% --- Executes during object creation,
after setting all properties.
function edit12_CreateFcn(hObject,
eventdata, handles)
% hObject  handle to edit12 (see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    empty - handles not
created until after all CreateFcns
called
% Hint: edit controls usually have a
white background on Windows.
%       See ISPC and COMPUTER.
if ispc &&
isequal(get(hObject,'BackgroundColor')
,
get(0,'defaultUicontrolBackgroundColor
'))
set(hObject,'BackgroundColor','white')
;
end

% --- Executes on button press in
pushbutton4.
function pushbutton4_Callback(hObject,
eventdata, handles)
% hObject    handle to pushbutton4
(see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
idx=get(handles.listbox2,'value')
alarme_global=getappdata(handles.pushb
utton1, 'alarme_global')
[ns ms]=size(alarme_global);
if  ns>0
    filename=alarme_global{idx,1}
 h=analise_criar_alarmest2(filename)
    waitfor(h)
else
    x=msgbox('Tem de carregar um
alarme!')
end

% --- Executes on button press in
pushbutton5.
```

```
function pushbutton5_Callback(hObject,
eventdata, handles)
% hObject    handle to pushbutton5
(see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
close

% --- Executes on button press in
togglebutton1.
function
togglebutton1_Callback(hObject,
eventdata, handles)
% hObject    handle to togglebutton1
(see GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns
toggle state of togglebutton1
%%%%%%Monitorizar %%%%%%%
alarme_global=getappdata(handles.pushb
utton1, 'alarme_global')
idx_alarme=1;
lista_eventos=[];
tic
load mmsit
toc
L={['Inicio de pesquisa às: '
datestr(now,31)]};
set(handles.listbox1,'string',L)
pause(0.01)
while
get(handles.togglebutton1,'value')
    %obter gdh relativo ao instante a
pesquisar alertas - fica guardado na
    %varival d
    if get(handles.checkbox1,'value')
        d=now;
    else
d=datenum(strtrim(get(handles.edit2,'s
tring')),'yyyy-mm-dd HH:MM:SS');
    end
dir_dados=getappdata(handles.pushbutto
n1,'dir_dados');
    dia=day(d);
    if dia<10
        t=[num2str(0) num2str(dia)];
    else
        t=num2str(dia);
    end
    [mes mes1]=month(d);
    ano=year(d);
    if
get(handles.checkbox2,'value')%MSSIS
AIS
        ficheiro=['F' t mes1
num2str(ano) '.mat'];
        str=[ dir_dados '\'
num2str(ano) '\' num2str(mes) '\F\'
ficheiro];
        f_mssis=str;
    else
        f_mssis=[];
    end
    if
get(handles.checkbox3,'value')%PT AIS
        ficheiro=['F_PT' t mes1
num2str(ano) '.mat'];
        str=[ dir_dados '\'
num2str(ano) '\' num2str(mes) '\F_PT\'
ficheiro];
        f_pt=str;
    else
        f_pt=[];
    end
    if exist(str,'file')
        %carregar ficheiro
        ok=0;
        try
L=get(handles.listbox1,'string');[n
lixo]=size(L);
            L{n+1,1}='A ler
ficheiro...';
set(handles.listbox1,'string',L)
            load(str)%carrega
foorpint_aom e footprint_n_aom --
matrizes grandes
            str
            ok=1
        catch ME
            report = getReport(ME);
            h=msgbox(report);
            waitfor(h)
        end
        if ok %ficheiro AIS bom para
ser lido
alarme=alarme_global{idx_alarme,2}
L=get(handles.listbox1,'string');[n
lixo]=size(L);
          L{n+1,1}=['A ler o alarme '
alarme.filename];
set(handles.listbox1,'string',L)
%---&&&&&&&&&&&&&&&&&&&&&&&&&&&----
           poligono=alarme.area{1,2}
           if ~isempty(poligono)
% ------------POLIGONO-------------
               lat=poligono{1,5};
               lon=poligono{1,6};
               if isempty(lat) ||
isempty(lon)
                   lat=poligono{1,3};
                   lon=poligono{1,4};
               end
               'esolha na area'
               alarme.area{1,3}
               if alarme.area{1,3}==1
posicoes=footprint_aom(inpolygon(footp
rint_aom(:,3),footprint_aom(:,4),lat,l
on),:);
posicoes1=footprint_n_aom(inpolygon(fo
otprint_n_aom(:,3),footprint_n_aom(:,4
),lat,lon),:);
               elseif
alarme.area{1,3}==2
posicoes=footprint_aom(inpolygon(footp
rint_aom(:,3),footprint_aom(:,4),lat,l
on)==0,:);
posicoes1=footprint_n_aom(inpolygon(fo
otprint_n_aom(:,3),footprint_n_aom(:,4
),lat,lon)==0,:);
               else
buf=nm2deg(alarme.area{1,4})
```

```
 [latb, lonb] = bufferm(lat, lon,
buf,'in');
lat1=latb(1:numel(latb)-numel(lat)-
1);lon1=lonb(1:numel(lonb)-numel(lon)-
1);
posicoes=footprint_aom(inpolygon(footp
rint_aom(:,3),footprint_aom(:,4),lat1,
lon1),:);
posicoes1=footprint_n_aom(inpolygon(fo
otprint_n_aom(:,3),footprint_n_aom(:,4
),lat1,lon1),:);
posicoes=posicoes(inpolygon(posicoes(:
,3),posicoes(:,4),lat,lon)==0,:);
posicoes1=posicoes1(inpolygon(posicoes
1(:,3),posicoes1(:,4),lat,lon)==0,:);
                end
           latmean=mean(lat(1:end-
1));lonmean=mean(lon(1:end-
1));%h1=plotm(latmean,lonmean,27,'+w')
;set(h1,'tag','pt','markersize',20);
posicoes(:,9)=deg2nm(distance([latmean
lonmean],posicoes(:,3:4)));
posicoes1(:,9)=deg2nm(distance([latmea
n lonmean],posicoes1(:,3:4)));
     posicoes=[posicoes; posicoes1]
    clear footprint_aom
footprint_n_aom
 % ------------POLIGONO-------------
                if ~isempty(poligono)
%set(handles.listbox1,'string','A
pesquisar navios...')
                    t = clock;
[eventos tempos
etiqueta]=fcn_pesquisar_alarme(handles
,alarme,posicoes,d,mmsit)
tempo_decorrido=etime(clock, t)
           if ~isempty(eventos)
lista_alarmes{idx_alarme,1}=eventos
fcn_mostrar_alarmes_mapa(handles,event
os,etiqueta,alarme, idx_alarme)
                        save eventos
                    end
                end
           else
           end
%---&&&&&&&&&&&&&&&&&&&&&&&&&&&&----
           [n m]=size(alarme_global);
           if idx_alarme<n
idx_alarme=idx_alarme+1;
           else
               idx_alarme=1;
           end
        else
        end
    else
    end
    pause(1)
    if
get(handles.togglebutton1,'value')==0
L=get(handles.listbox1,'string')
        [n lixo]=size(L)
L{n+1,1}='O utilizador parou a
pesquisa...';
set(handles.listbox1,'string',L)
    end
end
```

```
% --- Executes on button press in
checkbox2.
function checkbox2_Callback(hObject,
eventdata, handles)
% hObject    handle to checkbox2 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns
toggle state of checkbox2

% --- Executes on button press in
checkbox3.
function checkbox3_Callback(hObject,
eventdata, handles)
% hObject    handle to checkbox3 (see
GCBO)
% eventdata  reserved - to be defined
in a future version of MATLAB
% handles    structure with handles
and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns
toggle state of checkbox3
function [eventos tempos
etiqueta]=fcn_pesquisar_alarme(handles
,alarme,posicoes,d,mmsit)
eventos=[]
tempos=[]
etiqueta=[]
if alarme.gdh{1,1}
    'a pesquisar gdh...'
posicoes=fcn_alarme_pesquisar_gdh(posi
coes,alarme,d);
    if alarme.navio{1,1}==1
        'a pesquisar navio...'
        [eventos_t
posicoes]=fcn_alarme_pesquisar_navio(p
osicoes,alarme,d,mmsit)
        vel_med=mean(posicoes(:,6))
        [n_contactos
~]=size(eventos_t)
        if alarme.metoc{1,1}==1
[eventos_t]=fcn_alarme_pesquisar_metoc
(eventos_t,alarme);
        else

        end
    else

    end

else
    if alarme.navio{1,1}==1
        'a pesquisar navio...'
        [eventos_t
posicoes]=fcn_alarme_pesquisar_navio(p
osicoes,alarme,d,mmsit);
        if alarme.metoc{1,1}==1
[eventos_t]=fcn_alarme_pesquisar_metoc
(eventos_t,alarme);
        else
        end
    else
    end
end
if ~isempty(eventos_t)
```

```
    eventos=eventos_t;
end
function
fcn_mostrar_alarmes_mapa(handles,event
os,etiqueta,alarme,idx_alarme)
idx=findobj('tag',num2str(idx_alarme))
;
if ~isempty(idx)
    delete(idx)
end
pause(0.01)
[n m]=size(eventos);
if alarme.prioridade==1
    cor='r';cor2='w';
elseif alarme.prioridade==2
    cor=[1 0.694 0.392];cor2='w';
elseif alarme.prioridade==3
    cor='y';cor2='r';
elseif alarme.prioridade==4
    cor='g';cor2='k';
else
    cor='w';cor2='b';
end
for i=1:n
h=plotm(eventos(i,3),eventos(i,4),'or'
,'markersize',12,'markeredgecolor',cor
2,'markerfacecolor',cor);set(h,'tag',n
um2str(idx_alarme),'buttondownfcn',{@f
cn_menu_alarme,handles})
    set(h,'userdata',eventos(i,:))
end
function
posicoes=fcn_alarme_pesquisar_gdh(posi
coes,alarme,d)
if isempty(posicoes)
else
    %%%%------------GDH-------------
    val_ini=alarme.gdh{1,2}{1,2}
    val_fim=alarme.gdh{1,2}{1,3}
    if d < val_ini || d > val_fim

        'olferferf'
    else
        %--PERIODO HORARIO
        per_hora=datestr(d,'HH');
        per_minutos=datestr(d,'MM');
        per=str2double([per_hora
per_minutos])
per_ini=str2double(alarme.gdh{1,2}{2,2
})*100
per_fim=str2double(alarme.gdh{1,2}{2,3
})*100
  if per < per_ini || per > per_fim
            posicoes=[];
        else
            [n m]=size(posicoes);
            per=zeros(n,1);
            for i=1:n
per_hora=datestr(posicoes(i,2),'HH');
per_minutos=datestr(posicoes(i,2),'MM'
);
per(i)=str2double([per_hora
per_minutos]);
            end
posicoes=posicoes(per>=per_ini &
per<=per_fim,:);
save treta00 posicoes per per_ini
per_fim
        end
    end
end
function [eventos
posicoes]=fcn_alarme_pesquisar_navio(p
osicoes,alarme,d,mmsit)
eventos=[];
if isempty(posicoes)
else
    if alarme.navio{1,1}==1%uso
condição logica
tipo_de_lista=alarme.navio{1,2}
        if alarme.navio{1,2}==1%lista
de navios
mmsi=cell2mat(alarme.navio{1,3});
idx=find(ismember(posicoes(:,1),mmsi(:
,1)));
            if ~isempty(idx)
posicoes=posicoes(idx,:);
 if alarme.navio{1,5}(1)==1%filtro
pelo SOG
posicoes=posicoes(posicoes(:,6)>=alarm
e.navio{1,5}(2) &
posicoes(:,6)<=alarme.navio{1,5}(3),:)
;
                else
     %naõ tem filtro SOG
                end
  if alarme.navio{1,6}(1)==1%filtro
pelo COG
   %Retirar rumo e variação
introduzidas pelo utilizador
rumo=alarme.navio{1,6}(2)
var=alarme.navio{1,6}(3)
    %Conversor rumo (geográfico para
trigonométrico)
                    rumo=450-rumo;
                    if rumo>=360
                        rumo=rumo-360;
                    end
         %Início da filtragem
 if rumo-var<0
idx=find((360+rumo-var<=posicoes(:,7)
& ...
posicoes(:,7)<=360) |
(rumo<=posicoes(:,7) & ...
posicoes(:,7)<=rumo+var)
|posicoes(:,7)==0);
    elseif rumo+var>360
idx=find((rumo<=posicoes(:,7) & ...
posicoes(:,7)<=360) |
(0<=posicoes(:,7) & ...
posicoes(:,7)<=rumo+var-
360)|posicoes(:,7)==0);
     else
idx=find(posicoes(:,7)>=(rumo-var) &
posicoes(:,7)<=rumo+var);
                    end
posicoes=posicoes(idx,:);
                else
         %nao tem filtro COG
                end
            else
            end

        else %lista indiferenciada
tipo_navio=alarme.navio{1,4}
```

```
if tipo_navio==1%todos os tipos de
navios
   if alarme.navio{1,5}(1)==1%filtro
pelo SOG
posicoes=posicoes(posicoes(:,6)>=alarm
e.navio{1,5}(2) &
posicoes(:,6)<=alarme.navio{1,5}(3),:)
;
                else
        %naõ tem filtro SOG
                end
 if alarme.navio{1,6}(1)==1%filtro
pelo COG
    %Retirar rumo e variação
introduzidas pelo utilizador
rumo=alarme.navio{1,6}(2)
var=alarme.navio{1,6}(3)
    %Conversor rumo (geográfico para
trigonométrico)
                    rumo=450-rumo;
                    if rumo>=360
                        rumo=rumo-360;
                    end
         %Início da filtragem
                    if rumo-var<0
idx=find((360+rumo-var<=posicoes(:,7)
& ...
posicoes(:,7)<=360) |
(rumo<=posicoes(:,7) & ...
posicoes(:,7)<=rumo+var)
|posicoes(:,7)==0);
          elseif rumo+var>360
idx=find((rumo<=posicoes(:,7) & ...
posicoes(:,7)<=360) |
(0<=posicoes(:,7) & ...
posicoes(:,7)<=rumo+var-
360)|posicoes(:,7)==0);
                    else
idx=find(posicoes(:,7)>=(rumo-var) &
posicoes(:,7)<=rumo+var);
                    end
posicoes=posicoes(idx,:);
                else
            %nao tem filtro COG
                end
     else%tipos de navios especificos
posicoes =
fcn_colocar_tipo(posicoes,mmsit);
        if tipo_navio==2%tanker
posicoes=posicoes(posicoes(:,9)>=80 &
posicoes(:,9)<=89,:);
    elseif tipo_navio==3%cargo
posicoes=posicoes(posicoes(:,9)>=70 &
posicoes(:,9)<=79,:);
  elseif tipo_navio==4%passenger
posicoes=posicoes(posicoes(:,9)>=60 &
posicoes(:,9)<=69,:);
 elseif tipo_navio==5%lawenforcement
posicoes=posicoes(posicoes(:,9)>=55 &
posicoes(:,9)<=57,:);
elseif tipo_navio==6
posicoes=posicoes(posicoes(:,9)==30,:)
;
       elseif tipo_navio==7
posicoes=posicoes(posicoes(:,9)==36,:)
;
                elseif tipo_navio==9
posicoes=posicoes(posicoes(:,9)>=79 &
posicoes(:,9)<=89,:);
                elseif tipo_navio==10
posicoes=posicoes(posicoes(:,9)>=60 &
posicoes(:,9)<=89,:);
                else
idx1=[30,36,55:57,60:89];
posicoes=posicoes(posicoes(:,9)~=idx1,
:);
                end
                save treta2 posicoes
                if ~isempty(posicoes)
alarme.navio{1,5}(2)

alarme.navio{1,5}(3)
  if alarme.navio{1,5}(1)==1%filtro
pelo SOG
posicoes=posicoes(posicoes(:,6)>=alarm
e.navio{1,5}(2) &
posicoes(:,6)<=alarme.navio{1,5}(3),:)
;
                    else
              %naõ tem filtro SOG
                    end
if alarme.navio{1,6}(1)==1%filtro pelo
COG
     %Retirar rumo e variação
introduzidas pelo utilizador
rumo=alarme.navio{1,6}(2)
var=alarme.navio{1,6}(3)
%Conversor rumo (geográfico para
trigonométrico)
                        rumo=450-rumo;
                        if rumo>=360
                       rumo=rumo-360;
                        end
             %Início da filtragem
                        if rumo-var<0
idx=find((360+rumo-var<=posicoes(:,7)
& ...
posicoes(:,7)<=360) |
(rumo<=posicoes(:,7) & ...
posicoes(:,7)<=rumo+var)
|posicoes(:,7)==0);
              elseif rumo+var>360
idx=find((rumo<=posicoes(:,7) & ...
posicoes(:,7)<=360) |
(0<=posicoes(:,7) & ...
posicoes(:,7)<=rumo+var-
360)|posicoes(:,7)==0);
                        else
idx=find(posicoes(:,7)>=(rumo-var) &
posicoes(:,7)<=rumo+var);
                        end
posicoes=posicoes(idx,:);
                    else
                    end
                else
                end
            end
        end
        if ~isempty(posicoes)
lista=unique(posicoes(:,1));
            [n m]=size(lista);
            for i=1:n
idx=find(posicoes(:,1)==lista(i));
                v=posicoes(idx,:);
                v=sortrows(v,2);
```

```
                if i==1
                    eventos=v(end,:);
                else
eventos(i,:)=v(end,:);
                end
            end
        end
else%nao se usa a condição logica
sobre os navios
    end
end
function [eventos_metoc
posicoes]=fcn_alarme_pesquisar_metoc(p
osicoes,alarme)
eventos_metoc=[];
posicoes=[];
function
eventos=fcn_alarme_comparar_eventos(ev
entos,eventos_t)
if isempty(eventos)
    eventos=eventos_t;
else
end
function fcn_menu_alarme(src,
eventdata,handles)
b = get(gcf,'selectiontype');
if strcmpi(b,'normal')
elseif strcmpi(b,'alt')
    'Right click'
    a=get(gco,'userdata');
    hcmenu = uicontextmenu;
    item1 = uimenu(hcmenu, 'Label',
['MMSI: '
num2str(a(1,1))],'foregroundcolor','b'
);
    item2 = uimenu(hcmenu, 'Label',
['GDH: ' datestr(a(1,2),31)]);
    item3 = uimenu(hcmenu, 'Label',
['LAT: ' num2str(a(1,3))]);
    item4 = uimenu(hcmenu, 'Label',
['LON: ' num2str(a(1,4))]);
    item5 = uimenu(hcmenu, 'Label',
['SOG: ' num2str(a(1,6)) ' Kts']);
    item6 = uimenu(hcmenu, 'Label',
['COG: ' num2str(a(1,7)) '°']);
    item7 = uimenu(hcmenu, 'Label',
'Marine
Traffic','foregroundcolor',[0.078,0.16
9,0.549],'Separator','On','Callback',
{@fcn_mt,handles,a});
    item8 = uimenu(hcmenu, 'Label',
'Análise da
Trajectória','foregroundcolor',[0.078,
0.169,0.549],'Callback',
{@fcn_at,handles,a});
    set(gco,'uicontextmenu', hcmenu)
else
    'Careful there, crazy man!'
end
function posicoes =
fcn_colocar_tipo(posicoes,mmsit)
[n m ]=size(posicoes);
if n>0
    for i=1:n
idx=find(mmsit==posicoes(i,1));
        if ~isempty(idx)
posicoes(i,9)=mmsit(idx,2);
        else
            posicoes(i,9)=0;
        end
    end
end
function fcn_mt(src,
eventdata,handles,b)
mmsi=b(1,1)
dir_trabalho=getappdata(handles.pushbu
tton1,'dir_trabalho');
a{1}='<html>';
a{2}='<head>';
a{3}=['<meta http-equiv="refresh"
content="0;url=http://www.marinetraffi
c.com/ais/pt/shipdetails.aspx?mmsi='
num2str(mmsi) '" />'];
a{4}='</head>';
a{5}='<body>teste';
a{6}='</body>';
a{7}='</html>';
fid=fopen([dir_trabalho
'teste.html'],'wt');
for i=1:7
    fprintf(fid, '%s\n',a{i});
end
fclose(fid);
winopen([dir_trabalho 'teste.html'])
function fcn_at(src,
eventdata,handles,b)
mmsi=num2str(b(1));
data = floor(b(2));
analise_trajectoria_metoc(mmsi,data)
```