M-GIS — Sistema Móvel Interoperável de Informação Geográfica

Jorge Cardoso[1], Artur Rocha[1], João Correia Lopes[2]

[1] INESC Porto, R. Dr. Roberto Frias, 4200-465 Porto.
http://www.inescporto.pt/
{jcsc,artur.rocha,jlopes}@inescporto.pt
[2] Faculdade de Engenharia da Universidade do Porto, R. Dr. Roberto Frias, 4200-465 Porto.
http://www.fe.up.pt/∼jlopes/
jlopes@fe.up.pt

Resumo Este artigo descreve uma forma de acesso interoperável a um sistema de informação geográfica através de dispositivos móveis de características limitadas (e.g. telemóveis ou *Personal Digital Assistants*). O sistema M-GIS segue uma arquitectura cliente-servidor tendo por base informação geográfica no formato GML que é transformada, através de XSLT, para o formato gráfico SVG. A informação geográfica em GML é servida por um *Web Feature Server*, o que permite que seja acedida de uma forma normalizada, independentemente do seu formato ou localização física, desde que seja respeitada a conformidade com a especificação. A aplicação cliente foi desenvolvida usando a tecnologia *Java Mobile Information Device Profile*. Os resultados obtidos indicam que é viável desenvolver um sistema móvel de visualização de informação geográfica recorrendo a normas e formatos abertos, com algumas limitações. As principais limitações do sistema estão relacionadas com a quantidade de informação que o cliente consegue, nesta data, processar.

1 Introdução

A ubiquidade dos dispositivos móveis como telemóveis e *PDAs* (*Personal Digital Assistants*) tem levado a uma crescente oferta de serviços orientados para essas novas plataformas. Um tipo de serviço muito útil para os utilizadores desses dispositivos poderá ser um serviço baseado em informação geográfica. A possibilidade de um utilizador consultar o seu telemóvel para obter o mapa da zona onde se encontra, ou para procurar uma determinada rua ou edifício e visualizar graficamente a sua localização é uma mais valia óbvia.

Existem já alguns sistemas deste género disponíveis, tais como o ArcPad [ESR03a] da ESRI, o GeoGIS [Geo03] da Geo InSight e o PocketGIS [Poc03] da Pocket Systems. Todos estes sistemas foram desenvolvidos para *PDAs* (não funcionam em telemóveis) e são descritos na secção 2. O problema destes sistemas é a interoperabilidade com outros sistemas de informação geográfica, uma vez que, regra geral, se fica limitado ao uso de informação geográfica num determinado formato gerada por sistemas de informação geográfica da mesma entidade. Uma vez que as bases de dados geográficos utilizadas são proprietárias, estes sistemas são, regra geral, incompatíveis com outros sistemas de informação. Isto significa que não podemos ter uma gestão descentralizada da informação geográfica utilizada pelo sistema M-GIS (a não ser que todos os centros utilizem a mesma tecnologia). Por outro lado, muitos destes sistemas são orientados para determinados dispositivos, i.e., apenas funcionam em *PDAs* com o sistema operativo *Pocket PC*, por exemplo.

O objectivo deste trabalho é fornecer uma arquitectura para sistemas móveis de informação geográfica independente de tecnologias proprietárias (nomeadamente, da tecnologia da base de dados geográfica) de forma a propiciar a sua potencial integração com diversos sistemas de informação geográfica.

O sistema *M-GIS — Mobile Geographic Information System* [Car03] utiliza informação geográfica em formato GML (*Geography Markup Language*) proveniente de um WFS (*Web Feature Server*) para a construção de mapas pesquisáveis em formato SVG (*Scallable Vector Graphics*). A programação do lado do cliente foi feita em J2ME (*Java 2 Micro Edition*), mais concretamente usando o perfil MID (*Mobile Information Device*).

O resto deste artigo está estruturado da seguinte forma: a secção 2 apresenta, de forma resumida, alguns sistemas de informação geográfica para dispositivos móveis; a secção 3 apresenta algumas das tecnologias mais relevantes para o desenvolvimento deste projecto; a secção 4 apresenta a arquitectura do projecto M-GIS; na secção 5 são apresentados alguns detalhes de implementação, assim como alguns testes realizados sobre o sistema e respectivos resultados; por fim, na secção 6, apresentamos algumas conclusões que se podem retirar do trabalho desenvolvido e apontamos o caminho para trabalho futuro.

2 GIS para dispositivos móveis

Nesta secção serão introduzidos, de forma resumida, alguns sistemas de informação geográfica para dispositivos móveis existentes.

ArcPad

O sistema ArcPad [ESR03a] é um produto comercial da ESRI [ESR03b] para *PDAs*. É um sistema de informação geográfico completo com capacidade para adicionar dados a partir da Internet através de tecnologia *wireless*; "navegar" nos mapas: aproximar e deslocar a vista, estabelecer *bookmarks* e centrar na posição GPS actual; e pesquisar os dados: localizar e identificar objectos geográficos. O ArcPad funciona em *PDAs* com sistema operativo Windows CE/Pocket PC. O sistema suporta dados geográficos no formato *shapefile*[1] e pode ser usado como cliente de um servidor ArcIMS.

Uma vez que este sistema apenas foi desenvolvido para a plataforma Windows CE/Pocket PC e implica a utilização de bases de dados geográficas proprietárias da ESRI, não vai de encontro aos objectivos propostos para o nosso trabalho.

GeoGIS

O GeoGIS [Geo03] é um sistema de informação geográfico para *PDAs* com o sistema operativo Palm OS. Esta aplicação utiliza dados geográficos convertidos de ficheiros *shapefiles* criados com o sistema ArcView da ESRI. Este sistema tem, entre outras, as seguintes funcionalidades: criação de projectos com temas (ou camadas de mapas) únicos a cada projecto; aproximar, afastar e deslocar a vista sobre o mapa; pesquisar o mapa; adicionar ou apagar objectos do tema activo.

O GeoGIS utiliza dados convertidos do formato *shapefile* da ESRI e apenas pode ser usado em dispositivos com o sistema operativo Palm OS. Para além disso, este sistema não usa uma arquitectura cliente-servidor, i.e., os dados geográficos têm de ser carregados manualmente para o dispositivo. Por estas razões, o sistema GeoGIS não satisfaz os requisitos definidos neste projecto.

PocketGIS

O PocketGIS [Poc03] é um sistema para dispositivos com o sistema operativo Windows CE. O sistema é composto por uma aplicação que corre no dispositivo móvel e por uma aplicação de *desktop*, o PocketGIS Connection, que serve para transferir os dados entre o PocketGIS e o *desktop* numa variedade de formatos: NTF, *Shapefile*, MIF (MapInfo), DXF, CSV, Raster (BMP and TIFF). A aplicação PocketGIS tem as seguintes funcionalidades: identificação de objectos do mapa; medição de distâncias; modificação da geometria de um objecto; alteração dos atributos de objectos. Para além disso, um sistema de GPS pode ser integrado com o PocketGIS.

O sistema PocketGIS suporta dados de diversos formatos através de um utilitário de *desktop* que converte os dados de vários formatos para o formato utilizado pelo PocketGIS (provavelmente um formato proprietário). No entanto, este sistema implica o carregamento manual da informação geográfica para o dispositivo. Para além disso, apenas funciona em dispositivos com o sistema operativo Windows CE, por isso não cumpre os requisitos do projecto.

[1] Os ficheiros do tipo *shapefile* são um formato de informação geográfica criado pela ESRI e aceites como *de facto standard* pela indústria dos SIG.

3 Tecnologias Relevantes

Nesta secção são apresentadas algumas das tecnologias mais relevantes para o desenvolvimento deste projecto.

3.1 *Geography Markup Language*

O GML [OGC01] é uma especificação desenvolvida pelo consórcio internacional OpenGIS. O grande objectivo do GML é fornecer uma plataforma neutra e aberta para a definição de objectos e esquemas geo-espaciais. O GML não é uma linguagem rígida mas antes um sistema que suporta a definição de linguagens de descrição geográfica. Posto muito simplesmente, o GML consiste num conjunto de esquemas (*schemas*) XML que podem ser estendidos de forma a se adaptarem a situações específicas, mas mantendo sempre uma base comum.

O GML está desenhado de forma a suportar a interoperabilidade e fá-lo através da definição de elementos geométricos básicos (todos os sistemas que suportam GML usam os mesmos elementos geométricos), de um modelo de dados comum e de um mecanismo para criação e partilha de esquemas (*schemas*) aplicacionais. A maior parte das comunidades de informação facilita a interoperabilidade dos seus sistemas publicando os esquemas GML definidos por si. Esta norma é assim fundamental para satisfazer os objectivos deste trabalho relacionados com a interoperabilidade e independência em relação a formatos proprietários

3.2 *Web Feature Services* e *Web Map Services*

Web Feature Services (WFS) e *Web Map Services* (WFS) são duas especificações do consórcio OpenGIS [OGC03].

O WMS [OGC02b] especifica o serviço de *Web Map*, ou seja, especifica a interface *Web* de um sistema que produz mapas (representações visuais de informação geográfica). A especificação define os tipos de pedidos e respostas que um serviço deste género deve suportar. A especificação WMS define três operações: *GetCapabilities*, que retorna meta-informação sobre o serviço; *GetMap*, que retorna uma imagem cujo conteúdo e dimensão são definidos pelo cliente; *GetFeatureInfo* (opcional), que retorna informação sobre objectos particulares mostrados no mapa.

O WFS [OGC02a] especifica o serviço de *Web Features*, ou seja, define a interface de um sistema que produz informação geográfica no formato GML. Os pedidos ao WFS podem ser codificados de duas formas: em XML ou em pares parametro-valor. A especificação WFS define vários tipos de pedidos, e.g.,: *DescribeFeatureType*, gera um esquema com a descrição dos tipos de objectos servidos pela implementação do WFS; *GetFeature*, permite obter um conjunto de objectos geográficos no formato GML; *GetCapabilities*, gera um documento XML que define as "capacidades" do WFS, por exemplo, que camadas de informação estão disponíveis.

Estas duas especificações têm por objectivo a interoperabilidade de sistemas de informação geográfica, uma vez que permitem aceder a informação geográfica de uma forma normalizada, residente em qualquer sistema que esteja conforme com a especificação. Daí também o interesse que estas normas têm para o nosso projecto.

3.3 *Scalable Vector Graphics*

O SVG (*Scalable Vector Graphics*) é uma linguagem para descrever gráficos bidimensionais em XML. O SVG permite três tipos de objectos gráficos: formas gráficas vectoriais (e.g., polígonos), imagens e texto.

Os desenhos SVG podem ser interactivos e dinâmicos. Através de *scripting*, é possível manipular o DOM (*Document Object Model*) do SVG e ter acesso a todos os elementos, atributos e propriedades. Além disso, a norma define um conjunto de *event handlers* (e.g. `onmouseover` e `onclick`) que podem ser atribuídos a qualquer objecto gráfico do SVG.

Existem, neste momento, três perfis de SVG: o perfil *SVG Full* e os perfis móveis: *SVG Tiny* e *SVG Basic*. O perfil *Full* inclui todos os módulos da especificação SVG. Os perfis *Tiny* e *Basic* foram

definidos tendo como alvo pequenos dispositivos móveis, com limitações de recursos. O perfil *Tiny* é orientado para dispositivos muito limitados enquanto que o *Basic* é orientado para dispositivos ou pouco mais potentes. O perfil *Tiny* é um subconjunto do perfil *Basic*, sendo este um subconjunto do SVG 1.1.

O perfil que nos interessa mais para o nosso projecto é, obviamente, o perfil *SVG Tiny*, uma vez que o nosso sistema é orientado para dispositivos muito limitados.

Sendo tanto o SVG como o GML linguagens XML, o primeiro é facilmente obtido através de aplicação de folhas de estilo XSLT, a partir do segundo.

4 Desenho do Sistema

A motivação principal deste projecto era desenvolver um sistema que permitisse visualizar informação geográfica, disponível num servidor em formato GML, em dispositivos móveis usando, na medida do possível, tecnologias e formatos abertos. O facto de o formato de entrada dos dados geográficos ser o GML iria permitir desenvolver um sistema independente da tecnologia de base de dados geográficos e, por isso mesmo, adaptável a vários sistemas.

A primeira versão do sistema utilizava ficheiros com dados GML como fonte de dados. No entanto esta solução era pouco flexívelpelo que foi desenvolvida uma segunda versão que tem como fonte de dados um servidor WFS. A introdução de um *Web Feature Server* tornou o sistema mais flexível uma vez que podem ser adicionadas ou retiradas fontes de dados geográficos de uma forma transparente para o sistema M-GIS. Para além disso, o uso de um WFS permite configurar o sistema com fontes de dados geográficos em diversos formatos incluindo formatos proprietários.

A arquitectura do sistema é apresentada na Figura 1. A informação geográfica é obtida com recurso a um WFS que, por sua vez, pode ser configurado de forma a obter os dados geográficos de diversas fontes, inclusive de outros WFS. O WFS devolve informação geográfica no formato GML. A informação geográfica em formato GML é transformada no formato gráfico SVG usando uma folha de transformação XSLT. O mapa, no formato SVG, é enviado ao cliente que o poderá manipular directamente.

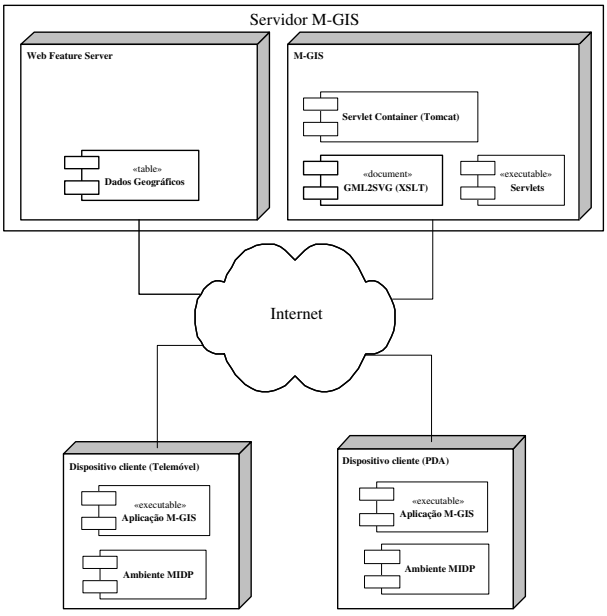


Figura 1: Arquitectura geral do sistema M-GIS

O Servidor M-GIS

O servidor responde a dois tipos de pedidos:

- Pedidos de listagem das camadas (*layers*) disponíveis no WFS.
- Pedido do conteúdo de uma área. Este pedido deve incluir a delimitação da área pedida.

A informação geográfica pode ser organizada por camadas, de forma a que o cliente apenas visualize o tipo de informação que necessitar. A configuração das camadas disponibilizadas é feita no WFS.

Quando o servidor recebe um pedido do conteúdo de uma área, esse pedido é redireccionado ao WFS (com algumas modificações de sintaxe). O WFS devolve a informação geográfica da área pedida, em formato GML, que o servidor M-GIS se encarregará de transformar para SVG, usando uma folha de transformação XSLT, e enviar ao cliente.

A comunicação entre o servidor e os clientes é feita sobre HTTP, sendo os pedidos dos clientes efectuados através do método HTTP GET codificados em pares parametro-valor no URL do pedido.

O Cliente M-GIS

O cliente M-GIS é uma aplicação Java[2] com as seguintes funcionalidades:

- Escolher a área a visualizar
- Escolher quais as camadas do mapa que se pretendem visualizar.
- Visualizar o mapa: deslocar, aproximar e afastar a vista.
- Pesquisar no mapa.
- Configurar o endereço do servidor que fornece os mapas.

A Figura 1 ilustra uma sequência de interacção do utilizador com o sistema M-GIS a partir de um telemóvel. A interacção resume-se a escolher a área a ser visualizada, as camadas de informação e, por fim, a navegação pelo mapa.

Nesta aplicação, o utilizador escolhe previamente a área que pretende visualizar sobre um mapa já presente no dispositivo. Depois de escolher a área que se pretende visualizar, a aplicação exibe uma lista das camadas de informação presentes no WFS. A lista das camadas disponibilizadas é obtida através de um pedido ao servidor M-GIS, que por sua vez fará o pedido ao WFS. Depois de escolhidas as camadas a visualizar é apresentado o mapa correspondente. A aplicação permite realizar operações de aproximação, afastamento e deslocamento da vista sobre o mapa. Permite também efectuar pesquisas e identificação de objectos no mapa. Todas estas operações são realizadas localmente no dispositivo, i.e., não existe comunicação com o servidor.

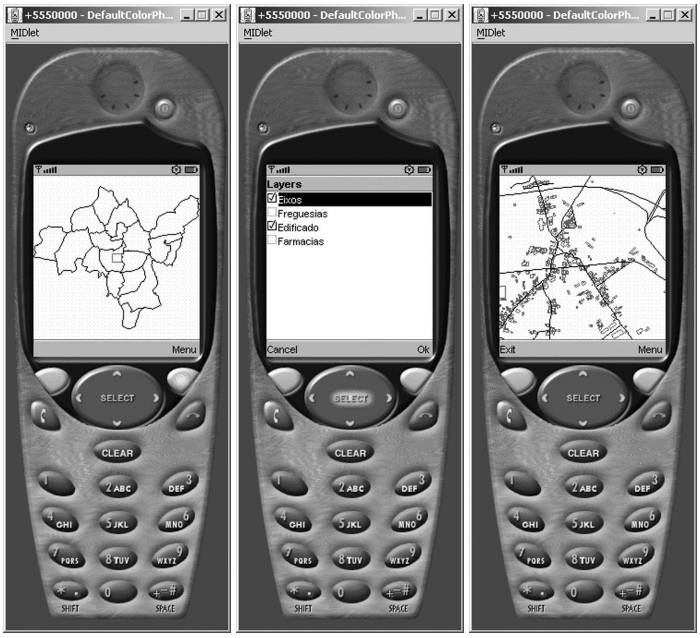
O núcleo central da aplicação do cliente é constituído por um interpretador e um visualizador de SVG desenvolvidos neste trabalho.

5 Implementação e Avaliação do Projecto

O servidor foi desenvolvido tendo por base o *servlet container Apache Tomcat* e consiste num conjunto de *servlets* e uma folha de transformação XSLT (transformação de GML para SVG Tiny).

O servidor é composto por duas *servlets*: `ListLayer` e `GetLayer`. A *servlet* `ListLayer` retorna ao cliente a lista de camadas (*layers*) de informação disponíveis no WFS. Esta lista é pedida directamente ao WFS, fazendo um pedido de *GetCapabilities*, que retorna, entre outras informações, as camadas configuradas no WFS. O resultado deste pedido (todos os pedidos e respostas ao WFS são feitos em XML) é interpretado e enviado ao cliente. A *servlet* `GetLayer` é responsável por obter do WFS a informação geográfica em GML representativa de uma determinada área (a área é definida em parâmetros da *servlet*), transformá-lo em SVG e enviar a resposta ao cliente. Os parâmetros aceites por esta *servlet* são os seguintes:

[2] A aplicação foi implementada usando a plataforma MIDP (*Mobile Information Device Profile*). Esta plataforma consiste, basicamente, num conjunto de APIs desenhadas especificamente para dispositivos muito limitados. É esta a plataforma encontrada nos telemóveis com tecnologia Java.



(a) Ecrã inicial (b) Camadas disponíveis (c) O mapa

Figura 1. Sequência de interacção com o sistema m-GIS

xmin A menor coordenada x do rectângulo que representa a área que se pretende visualizar.
xmax A maior coordenada x do rectângulo que representa a área que se pretende visualizar.
ymin A menor coordenada y do rectângulo que representa a área que se pretende visualizar.
ymax A maior coordenada y do rectângulo que representa a área que se pretende visualizar.
layers Uma lista de camadas que se pretende visualizar. Os nomes das camadas são separados por vírgulas.

Estes parâmetros são encapsulados num pedido do tipo *GetFeature* que é feito ao WFS. O resultado deste pedido ao WFS é um documento GML que será depois transformado em SVG e devolvido ao cliente. A transformação é feita recorrendo à API JAXP da Sun e a uma folha de transformação XSLT. Originalmente foi usado o processador de XML *Xerces*, uma vez que é este o processador usado pelo J2SDK1.4.x. No entanto, devido à existência de alguns *bugs* optou-se por usar o processador Saxon 6.5.2 [Mic03a]. Esta *servlet* mantém a folha de transformação XSLT em *cache* de forma a diminuir o tempo de interpretação do ficheiro XSLT.

Neste projecto foi usado o *deegree Web Feature Server* [Dee03] como WFS.

Transformação GML para SVG

A transformação de documentos GML para documentos SVG é feita através de uma folha de transformação (XSLT).

A folha de transformação usada para transformar GML em SVG foi adaptada de um exemplo da *OS MasterMap* [Mas03]. Foram introduzidas algumas modificações básicas, uma vez que a XSLT original se destinava a transformar GML em SVG Full e não em SVG Tiny. Uma dessas modificações é a conversão dos números para a gama suportada pela norma SVG Tiny. Para além disso foi modificada a forma e o tipo de estilos aplicados aos elementos SVG, mais uma vez devido às restrições impostas pela norma SVG Tiny, tais como a de não suportar estilos CSS.

A norma SVG Tiny suporta apenas números de vírgula fixa entre -32767.9999 a 32767.9999. No entanto a informação geográfica, contida nos ficheiros GML, pode usar valores arbitrariamente grandes para representar coordenadas de pontos no espaço. Isto obriga a que a transformação de

GML para SVG efectue uma modificação de escala. Na folha de transformação utilizada essa alteração é feita recorrendo aos valores do elemento <gml:boundedBy>. Este elemento GML define uma caixa (*bounding box*) dentro da qual todos os pontos definidos no documento GML devem estar contidos. Assim podemos usar o maior valor absoluto das coordenadas dessa caixa, para definir um factor de escala a aplicar a todos os outros valores, de forma a que o resultado esteja dentro da gama pretendida.

Uma vez que queremos distinguir visualmente objectos geográficos diferentes é necessário aplicar diferentes estilos a esses objectos no documento SVG resultante da transformação. Para tal ser possível é necessário que o documento GML contenha informação sobre o tipo de objecto representado, i.e., é necessário que o documento GML indique se determinado objecto é uma estrada, um edifício, um jardim, etc. A norma GML não define elementos para representar objectos como os mencionados atrás, mas define regras para a construção de esquemas GML que definem esses elementos. Assim, foi usado um esquema GML[3] que define, entre outras coisas, elementos para representar edifícios, zonas verdes e vias. A folha de transformação reconhece os elementos definidos no esquema e aplica diferentes estilos conforme o tipo de objecto geográfico representado (e.g. os jardins são pintados a verde).

O Exemplo 1 mostra um exemplo de um documento GML típico, conforme o esquema desenvolvido. O resultado da transformação em SVG desse documento é apresentado no Exemplo 2.

Podemos ver que um objecto geográfico, por exemplo, uma rua, é representado em SVG usando um elemento <g>, sendo que o nome é codificado no elemento <title> e a geometria no elemento <path>.

O estilo aplicado aos objectos é diferenciado, agrupando objectos do mesmo tipo dentro de um elemento <g> e atribuindo a esse elemento o estilo definido para o tipo de objecto. Podemos ver pelo Exemplo 2 que o edifício está inserido num elemento <g> com uma cor castanha (`fill="#905050"`) enquanto que a rua não tem preenchimento (`fill="none"`).

O elemento <g> de topo serve apenas para realizar uma adaptação do eixo das coordenadas y. No SVG, o eixo y, tem uma orientação de cima para baixo, enquanto que no GML usado, o eixo y tem uma orientação de baixo para cima. Podemos ver que o elemento <g> de topo define uma transformação que inverte o eixo das coordenadas y.

Praticamente todos os elementos de geometria do GML são transformados usando o elemento <path> do SVG. A excepção é o elemento <Point> do GML que é convertido para o elemento <circle> do SVG. De facto, este elemento, <circle>, é apenas definido uma vez no ficheiro SVG e todas as ocorrências de um ponto no documento são transformadas em referências para esse elemento. Isto é feito definindo o elemento <circle> dentro do elemento <defs> do SVG e atribuindo um "`id`" ao elemento. O elemento <path> é usado para representar quase todos os elementos geométricos porque permite optimizar o documento SVG resultante em termos de tamanho final.

Interpretação de SVG

Para realizar a interpretação do documento SVG no cliente é utilizado um interpretador to tipo *pull* fornecido pela biblioteca kXML [Enh03]. Um exemplo do uso do kXML para realizar interpretação de XML pode ser encontrado em [Knu02].

Uma vez que o kXML é um interpretador de XML, e não um interpretador de SVG, é necessário construir, em cima do kXML, algo que interprete os elementos e atributos SVG. O Exemplo 3 mostra parte de um documento SVG constituído por apenas um polígono. Um polígono é especificado através do elemento <polygon>, sendo os pontos que constituem o polígono definidos no atributo "`points`", como uma sequência de pares de valores numéricos. Neste caso, o kXML é capaz de nos fornecer, por exemplo, o valor do atributo "`points`". No entanto para termos acesso aos pontos que constituem o polígono é necessário interpretar esse valor (para o kXML, trata-se apenas de uma cadeia de caracteres).

[3] O esquema GML usado foi adaptado de um já existente criado no âmbito de um outro projecto desenvolvido no INESC Porto ([Mon02]).

```
<?xml version="1.0" encoding="UTF-8"?>
<Mapa xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:gml="http://www.opengis.net/gml" xsi:noNamespaceSchemaLocation="mgismap.xsd">
    <gml:boundedBy>
        <gml:Box>
            <gml:coord>
                <gml:X>-36391.6369611</gml:X><gml:Y>170507.5437507</gml:Y>
            </gml:coord>
            <gml:coord>
                <gml:X>-36012.3401935</gml:X><gml:Y>170770.2912157</gml:Y>
            </gml:coord>
        </gml:Box>
    </gml:boundedBy>
    <MapaMember>
        <Edificado>
            <EdificadoMember>
                <Edificio fid="_1">
                    <gml:name>Edificio</gml:name>
                    <gml:coverage>
                        <gml:Polygon>
                            <gml:outerBoundaryIs>
                                <gml:LinearRing>
                                    <gml:coord>
                                        <gml:X>-36263.089</gml:X>
                                        <gml:Y>170724.85</gml:Y>
                                    </gml:coord>
                                    <!--  ...   -->
                                    <gml:coord>
                                        <gml:X>-36263.089</gml:X>
                                        <gml:Y>170724.85</gml:Y>
                                    </gml:coord>
                                </gml:LinearRing>
                            </gml:outerBoundaryIs>
                        </gml:Polygon>
                    </gml:coverage>
                </Edificio>
            </EdificadoMember>
        </Edificado>
    </MapaMember>
    <MapaMember>
        <EixosVia>
            <ViaMember>
                <Eixo fid="_237">
                    <gml:name>Avenida do Lidador da Maia</gml:name>
                    <!-- ... -->
                </Eixo>
            </ViaMember>
        </EixosVia>
    </MapaMember>
</Mapa>
```

Exemplo 1: Exemplo de um documento GML

O Exemplo 3, é um exemplo típico do conteúdo de um documento SVG. O que mostra que grande parte da informação está contida no valor dos atributos dos elementos que definem elementos gráficos. Devido a esta estrutura típica de um documento SVG, acontece que se torna ineficiente interpretar o SVG. A razão é, muito simplesmente, a seguinte: o documento SVG tem de ser lido praticamente duas vezes. Primeiro, o kXML tem de fazer a interpretação para devolver à aplicação os elementos e atributos; depois a aplicação tem de interpretar o conteúdo dos atributos (na maior parte dos casos extrair valores numéricos).

A alteração mais natural consiste em incluir no kXML funcionalidades de interpretação de documentos SVG. Ou seja, obrigar o kXML a interpretar alguns atributos e, em vez de devolver à aplicação uma cadeia de caracteres, devolver, por exemplo, um *array* de pontos. Desta forma a interpretação do ficheiro é feita de uma só vez. Esta alteração foi levada a cabo modificando-se o código fonte do kXML (o kXML é um projecto *open source*), de forma a interpretar os atributos "`points`" dos elementos <*polygon*> e <*polyline*> e o atributo "`d`" do elemento <*path*>. São estes os atributos responsáveis pela maior parte da informação nos nossos documentos SVG.

```
<?xml version="1.0" encoding="UTF-8"?>
<svg id="svgAll" width="100%" height="100%" preserveAspectRatio="xMidYMid meet"
    viewBox="-6982.9544 -32767.9999 72.7808 50.4169">
<defs>
    <g id="pointSymbol" overflow="visible">
        <circle cx="0" cy="0" r="1.0" fill="#000000" stroke="#000000"/>
    </g>
</defs>
<g transform="matrix(1 0 0 -1 0 0)" fill="none" stroke-width="0.1" stroke="#000000">
    <g fill="#905050" stroke="#000000" stroke-width="1">
        <g id="_1">
            <title>Edificio</title>
            <path d="M-6958.2882,32759.2805z"/>
        </g>
    </g>
    <g fill="none" stroke="#000000" stroke-width="1">
        <g id="_237">
            <title>Avenida do Lidador da Maia</title>
            <path d="M-6985.7329,32804.0571l159.0339,-46.9665"/>
        </g>
    </g>
</g>
</svg>
```

Exemplo 2: Resultado da transformação em SVG do documento do Exemplo 1

```
<g id="_N3HE1s3th50">
    <title>Pedroucos</title>
    <polygon points="-7057.0669,30810.5546 -7055.2255,30829.6328
-7052.8957,30837.5716 -7050.1427,30840.2181 -7046.4364,30843.9228 -7036.6944,30849.7447
-7030.9762,30852.0734 -7027.9053,30852.9202 -7026.4227,30857.2601 -7025.7875,30858.8479
-7023.6697,30858.5304 -7018.7986,30856.8367 -7016.9983,30856.8367 -7014.2451,30857.2601
-7009.6919,30857.8952 -7004.8207,30858.7421 -6999.9496,30859.483"/>
</g>
```

Exemplo 3: Parte de um documento SVG típico

Testes e Resultados

Esta secção apresenta os resultados de algumas medições efectuadas sobre o sistema M-GIS. O sistema é analisado segundo vários eixos: memória consumida, tamanho máximo do mapa visualizado e tempos de transformação e de interpretação dos documentos SVG.

Os testes foram efectuados usando um computador com processador AMD Athlon a 1 GHz e com 256 Mb de memória RAM. Neste computador foram instalados o servidor WFS e o servidor m-GIS. O emulador de telemóvel usado foi também executado nesta máquina. No caso do PDA, foi usado um HP Jornada 548 Pocket PC com 32Mb de memória e com a máquina virtual Personal Java [Mic03b] da Sun instalada. Neste caso não existe emulação. A MIDlet foi executada recorrendo à ferramenta ME4SE [ME403] que permite executar MIDlets em ambientes Java "normais". A ligação do PDA com o computador onde se encontrava o servidor foi feita através de USB.

Uma das características dos dispositivos MIDP é a quantidade de memória reservada para as aplicações Java. A quantidade de memória do dispositivo impõe um limite à quantidade de informação geográfica que pode ser visualizada pelo dispositivo, ou seja, limita o tamanho do mapa visualizado.

A Tabela 1 apresenta, para uma determinada quantidade de memória, o tamanho máximo que o dispositivo consegue visualizar. Estes valores foram obtidos com o emulador de telemóvel do WTK[4]. Apenas foram testados mapas com três tipos de informação: eixos de via, edificado e eixos de via mais edificado. Podemos ver que com 192Kb conseguimos visualizar mapas dos eixos de via de dimensão até cerca de 1x1 quilómetros. Para visualizar mapas de edificado precisamos de, pelo menos, 256Kb de memória.

Um dos factores mais limitativos do sistema M-GIS é o tempo de transformação e de interpretação dos documentos SVG. A Tabela 2 apresenta alguns tempos medidos com o emulador de telemóvel do WTK. A tabela apresenta a dimensão do mapa, as camadas presentes no mapa, o tempo que demora a transformação de GML para SVG, o tempo que o dispositivo demora a efectuar a

[4] *Wireless Toolkit* – O *toolkit* de desenvolvimento de aplicações MIDP da Sun.

Tabela 1 Dimensões máximas dos mapas de acordo com a memória disponível

Memória (Kb)	Dimensão do mapa (m)	Camadas	Tamanho do documento SVG (Kb)
192	1033x1029	eixos	29.1
256	1549x1544	eixos	55.0
320	2324x2316	eixos	95.5
384	2840x2831	eixos	138.2
448	3098x3088	eixos	159.7
512	3615x3603	eixos	205.2
256	258x257	edificado	22.7
320	516x515	edificado	65.0
448	775x772	edificado	103.4
256	258x257	eixos + edificado	34.7
384	516x515	eixos + edificado	78.3
448	775x772	eixos + edificado	124.3

interpretação do documento[5], o tempo que o mapa demora a ser desenhado no ecrã do dispositivo e o tamanho do documento SVG.

Tabela 2 Tempos de transformação e de interpretação

Dimensão mapa (m)	Camadas	Tempos (segundos)			Tam. SVG (Kb)
		XSLT	Interpretação	*Rendering*	
258x257	eixos	0.8	20.5	3.2	13.0
516x515	eixos	0.8	22.6	3.6	14.2
775x772	eixos	1.0	33.8	5.4	21.9
1033x1029	eixos	1.1	46.3	7.3	29.9
1291x1287	eixos	0.5	57.3	8.9	37.5
1549x1544	eixos	1.7	83.1	13.0	55.4
1807x1801	eixos	2.5	108.9	16.9	72.4
2066x2059	eixos	25.5	128.2	19.5	85.3
2324x2316	eixos	15.2	147.0	22.7	97.8
2582x2574	eixos	49.1	175.2	25.9	117.1
2840x2831	eixos	72.7	211.5	30.4	141.5
3098x3088	eixos	98.4	244.8	34.1	163.5
3357x3346	eixos	147.0	285.5	38.4	190.2
3615x3603	eixos	160.0	313.7	41.3	210.1
258x257	edificado	0.3	40.6	6.5	23.3
516x515	edificado	0.9	111.6	14.6	66.5
775x772	edificado	27.4	175.1	17.2	105.9
258x257	eixos+edificado	2.3	59.3	9.7	35.6
516x515	eixos+edificado	2.7	130.0	18.1	80.2
775x772	eixos+edificado	54.3	204.2	22.5	127.3

A Tabela 3 mostra alguns tempos medidos com um dispositivo real: um PDA HP Jornada.

Tabela 3 Tempos de de interpretação num dispositivo real: HP Jornada

Dimensão do mapa (m)	Camadas	Tempos (segundos)	
		Interpretação	*Rendering*
306x305	eixos+edificado	17.0	3.0
816x814	eixos+edificado	50.0	9.0

Os valores apresentados indicam que o sistema apenas responde de forma aceitável para mapas muito pequenos. Quando a área abrangida pelo mapa é grande, ou quando escolhemos visualizar várias camadas, o tempo de espera aumenta consideravelmente tornando o sistema muito pouco usável.

Basicamente, existem dois gargalos no sistema:

[5] Este tempo inclui também o tempo de ligação ao servidor, ou seja, é medido desde o momento em que é feito o pedido ao servidor até a interpretação do documento estar concluída.

– A transformação através de XSLT. Quando o documento GML se torna demasiado grande, a transformação para SVG torna-se demasiado lenta. Existem vários factores que contribuem para isto. A transformação é feita através de XSLT em Java usando a API JAXP. No nosso caso é usado o processador de XML Saxon. A transformação através de XSLT requer a criação das árvores DOM tanto do documento a transformar como do documento XSLT. À medida que o documento a transformar aumenta, também aumenta o tempo de interpretação do documento e da criação da respectiva DOM. Para além disso, as árvores DOM requerem muita memória, o que, no nosso caso vai obrigar ao uso de memória virtual durante a transformação e consequente degradação do desempenho.
– A interpretação e *renderização* do SVG no dispositivo. Uma vez que estamos a lidar com dispositivos muito limitados a nível de processamento, é natural que o desempenho na interpretação e *renderização* dos documentos SVG seja fraco.

6 Conclusões e Trabalho Futuro

O objectivo principal deste projecto era obter uma resposta à pergunta: "Será viável desenvolver um sistema móvel de visualização de informação geográfica tendo por objectivo a interoperabilidade de sistemas GIS, isto é, recorrendo a normas e formatos abertos?"

Os resultados obtidos indicam que sim, mas com algumas limitações. As principais limitações do sistema estão relacionadas com a quantidade de informação que o cliente consegue processar. Estas limitações são de duas ordens: a primeira diz respeito à memória necessária para visualizar um determinado mapa; a segunda relaciona-se com o tempo necessário para processar um mapa. Obviamente que estas são limitações de alguns dispositivos actuais. É de esperar que os próximos dispositivos melhorem o desempenho do sistema no que diz respeito a estas restrições.

Uma outra limitação do sistema prende-se com o aspecto gráfico. As APIs gráficas do MIDP são muito básicas pelo que apenas se conseguem produzir mapas muito simples graficamente. Uma possível forma de resolver este problema seria recorrer às APIs de alguns fabricantes de telemóveis, que disponibilizam funções gráficas um pouco mais completas. A desvantagem seria a redução da portabilidade da aplicação.

As limitações apontadas nos parágrafos anteriores dizem respeito ao cliente, no entanto, existem também algumas limitações, no sistema M-GIS, do lado do servidor. Estas limitações referem-se à transformação de GML para SVG. Contudo, uma vez que as opções que podemos tomar para a implementação do servidor são mais alargadas, estas limitações não causam muita preocupação. Seria perfeitamente possível, por exemplo, implementar a transformação de GML para SVG sem usar XSLT; a transformação pode ser feita usando *Perl*, ou mesmo Java, directamente sobre o documento GML. Obviamente, esta solução diminuiria a flexibilidade do sistema, mas poderia minorar o problema da eficiência da transformação.

Trabalho Futuro

O sistema desenvolvido pode evoluir principalmente segundo três eixos: arquitectura, desempenho e funcionalidades.

Arquitectura Em termos da arquitectura, podemos, por exemplo, pensar em adaptar o mapa resultante da transformação do GML em SVG às capacidades do dispositivo cliente. Deste modo os mapas gerados poderiam ser mais ou menos complexos dependendo da capacidade de processamento e/ou gráficas do dispositivo. Esta adaptação poderá ser realizada recorrendo a diferentes folhas de transformação XSLT (uma por cada tipo de dispositivo) e a um mecanismo de negociação entre o cliente e o servidor de forma a ser estabelecido qual o tipo de mapa suportado pelo cliente.

Desempenho Ao nível do desempenho, o sistema pode ser melhorado principalmente ao nível da transformação do GML para SVG. Para se saber a que nível actuar sobre esta transformação seria, no entanto, necessário um estudo mais aprofundado sobre as causas do baixo desempenho

do sistema a esse nível. A solução tanto pode passar por reescrever a folha de transformação de uma forma mais eficiente, como usar extensões do processador XSLT, ou até mesmo implementar a transformação sem recorrer a XSLT. Do lado do cliente o desempenho da aplicação está limitado ao desempenho do próprio dispositivo pelo que nesta vertente existe existe menos que possamos fazer.

Funcionalidades Quanto às funcionalidades do sistema, existem várias que podem ter interesse para o utilizador:

Localização actual Exibir o mapa da zona onde o utilizador se encontra no momento. Um serviço deste tipo tem interesse, por exemplo, no caso de pessoas de visita a cidades desconhecidas. Isto implica o uso de dispositivos com capacidade para identificar a localização corrente, o que, no caso dos telemóveis pode ser feito através da informação das células da rede.

Pontos de Interesse Lista de pontos de interesse na área visualizada pelo utilizador. Poderia ser acrescentado uma opção que listasse os pontos de interesse (possivelmente configurados pelo utilizador) da área visualizada, e.g., uma lista de monumentos, postos de bombeiros e de polícia, etc.

Pesquisa global Neste momento a pesquisa é efectuada sobre a área pedida pelo utilizador. Poder-se-ia implementar uma nova funcionalidade que permitisse ao utilizador, por exemplo, pesquisar por uma determinada rua em todo o concelho. O utilizador poderia, depois, visualizar o mapa da área em que situa a rua pretendida.

Referências

[Car03] Jorge C. S. Cardoso. M-GIS: Mobile Geographic Information System. Relatório do projecto, INESC Porto, 2003.

[Dee03] Deegree. Projecto Deegree, 2003. `http://www.deegree.org`.

[Enh03] Enhydra.org. Projecto kXML, 2003. `http://kxml.enhydra.org`.

[ESR03a] ESRI. ArcPad, 2003. `http://www.esri.com/software/arcpad/index.html`.

[ESR03b] ESRI. ESRI - *GIS and Mapping Software*, 2003. `http://www.esri.com`.

[Geo03] Geo InSight. GeoGIS, 2003. `http://www.geoinsight.com/Products/Mobile/GeoGIS.cfm`.

[Knu02] Jonathan Knudsen. *Parsing XML in J2ME[tm]*. `http://wireless.java.sun.com/midp/articles/parsingxml`, Março 2002.

[Mas03] OS MasterMap. *Style and XML Examples*, 2003. `http://www.ordnancesurvey.co.uk/os_mastermap/xml/`.

[ME403] ME4SE. ME4SE, 2003. `http://www.me4se.org`.

[Mic03a] Michael Kay. Saxon, 2003. `http://saxon.sourceforge.net/saxon6.5.2/`.

[Mic03b] Sun Microsystems. *Personal Java*, 2003. `http://java.sun.com/products/personaljava/`.

[Mon02] Anabela Soares Pinto Monteiro. Servidor de Mapas Vectoriais. Relatório do projecto, INESC Porto, 2002.

[OGC01] OGC. *Geography Markup Language (GML) 2.0*. Especificação, OGC, Fevereiro 2001.

[OGC02a] OGC. *Web Feature Service Implementation Specification*, 2002. `http://www.opengis.org`.

[OGC02b] OGC. *Web Map Service Implementation Specification*, 2002. `http://www.opengis.org`.

[OGC03] OGC. Consórcio *Open GIS*, 2003. `http://www.opengis.org`.

[Poc03] Pocket Systems Ltd. PocketGIS, 2003. `http://www.pocket.co.uk`.