UNIVERSIDADE FEDERAL DE PERNAMBUCO
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
CENTRO DE INFORMÁTICA

SOF – SOCIAL FRAMEWORK

FRAMEWORK PARA APLICAÇÕES SOCIAIS DE LARGA ABRANGÊNCIA

TRABALHO DE GRADUAÇÃO

Aluno: Renato Viana Ferreira (rvf[at]cin.ufpe.br)
Orientador: Carlos André Guimarães Ferraz (cagf[at]cin.ufpe.br)

AGOSTO DE 2007

“Gggggrrrrrr rrrraaaahhh rrrrrrggghhhhnn”
Chewbacca
Star Wars Episode V: The Empire Strikes Back
1980

Resumo

O potencial das aplicações sociais e sua capacidade de integrar pessoas ainda não são completamente explorados. A abrangência de televisores, aparelhos celulares e internet permite a criação de software com objetivo de motivar a população para solucionar seus problemas cotidianos, porém, ainda não existem ferramentas que auxiliem desenvolvedores de software na construção de aplicações deste tipo. Neste contexto, este projeto apresenta o Social Framework, que surge no intuito de preencher esta lacuna, provendo funcionalidades comuns para as aplicações sociais de larga abrangência. O Mosquito.Net, uma aplicação, que integrada ao SoF, busca auxiliar autoridades e população no combate à dengue também apresentada.

Palavras-chave: Aplicações Sociais, Framework, TV Digital

Abstract

The potential of social applications and their capacity to integrate people is not fully exploited yet. The broadness of televisions, mobile phones and internet enables the creation of software aiming to motivate the population to solve its daily problems, however there are not tools to aid developers to build applications of such kind. In this context, this project presents the Social Framework, which rises looking forward to fill this gap, providing common functionalities to the wide broadness social applications. The Mosquito.Net, an application that integrated with the SoF, aims to help authorities and the population to combat dengue fever is also presented.

Keywords: Social Software, Framework, Digital TV

A Darth Vader, o incompreendido.

Agradecimentos

À *FAST Soluções Tecnológicas*, pelo companheirismo e apoio.

Aos amigos, pela amizade verdadeira.

À minha família: Silvio, Sonia, Rebeca e vovó Inês, por me suportarem ao longo dos anos.

Ao meu orientador Carlos Ferraz, pela confiança e pelo tempo dedicado a este projeto.

E, finalmente, a Deus, por ser.

Verdadeiramente grato!

Sumário

Lista de Figuras

1 Introdução

A TV e dispositivos móveis, como celulares, estão presentes no cotidiano de uma grande parcela da sociedade atual. No Brasil, em 2004, 90% dos domicílios possuíam televisão [IBGE 2005] e em 2007 o país já possui mais de 100 milhões de celulares [IDG Now! 2007]. A partir da definição do Sistema Brasileiro de TV Digital [SBTVD 2007], possibilitando no Brasil a implantação da TV Digital Interativa terrestre e aberta, que já é uma realidade em inúmeros países, cria-se também um canal de retorno para a TV. Este canal permite o envio de informações do telespectador através de aplicações de TVD com poucos cliques no controle remoto, algo que é possível com os aparelhos celulares.

Através destes meios e também da internet, uma quantidade gigantesca de pessoas pode enviar e receber informações de forma bastante simples, porém essa imensa capacidade ainda não é completamente explorada por soluções de tecnologia, nem pelas chamadas aplicações sociais.

A possibilidade de obter informações vindas de todas as classes sociais cria oportunidades para aplicações usarem essa abrangência em prol da solução de problemas que atingem toda a sociedade. Esses podem afetar diversos aspectos das nossas vidas, como por exemplo: doenças infectocontagiosas, criminalidade, dentre outros.

Para a solução de problemas deste tipo, é fundamental o mapeamento dos locais de incidência e levantamento de recursos o que é extremamente custoso e praticamente impossível sem o apoio da população em geral [Lloyd 2003].

Neste contexto, a pró-atividade inerente a aplicações de TV Digital, permitida pelos celulares e também a necessidade de ver os seus problemas cotidianos solucionados, combinadas com ações efetivas de marketing, criaria na população o desejo de contribuir no envio de informações úteis e relevantes para otimizar a resolução de tais problemas e também na doação de recursos.

A utilização destas informações não se restringe a resolução de problemas sociais, mas também pode se aplicar a realização de censos, pesquisas de opinião, eleições, etc.

O projeto proposto surge como um framework, chamado de SoF – *Social Framework*, para permitir o desenvolvimento mais prático e ágil de aplicações sociais de larga abrangência. Este framework conterá os itens básicos para qualquer aplicação deste tipo, distribuídos em seus módulos.

1.1 Metodologia de Trabalho

A metodologia utilizada no desenvolvimento deste trabalho se constitui das seguintes fases:

- **Estudo e revisão bibliográfica sobre frameworks e seu processo de desenvolvimento** – inicialmente buscou-se entender o conceito e de que é composto um framework. Foi necessário também compreender as etapas no desenvolvimento deste tipo de software, assim como suas vantagens e desvantagens.

- **Análise e revisão bibliográfica das aplicações sociais** – foi realizado o estudo do conceito de aplicações sociais e os tipos atualmente existentes. Vários exemplos foram observados.

- **Proposição de nova abordagem para aplicações sociais** – como a abordagem dada por este projeto às aplicações sociais é inovadora, se torna necessário a proposição da idéia, assim como a análise de possíveis aplicações existentes que possam se encaixar nessa categoria. Também é interessante a proposição de um modelo de negócio simples.

- **Escolha e levantamento dos requisitos das aplicações sociais exemplos** – após o estudo do processo de desenvolvimento de frameworks, levanta-se a necessidade de aplicações exemplos que direcionem o desenvolvimento do SoF.

- **Requisitos, análise, projeto, implementação e testes do SoF** – seguindo esta escolha das aplicações exemplos, foram elicitados os requisitos referentes ao SoF, a partir deles foram realizadas as atividades de análise e projeto, baseadas no estudo de padrões de projeto. Por fim o Social Framework foi desenvolvido e testado.

- **Desenvolvimento da aplicação de testes** – finalmente para validar a implementação do SoF, foi desenvolvida uma aplicação de teste. A aplicação escolhida para ser desenvolvida, foi o Mosquito.Net.

1.2 Organização do Trabalho de Graduação

Este trabalho de graduação está organizado em 6 capítulos. O segundo capítulo apresenta ao leitor o conceito de frameworks, assim como seu processo de desenvolvimento e suas classificações. O terceiro capítulo foca nas aplicações sociais e suas categorias, apresentando a nova abordagem proposta por este trabalho. No capítulo 4 é apresentado o SoF – Social Framework em todas as fases do seu desenvolvimento. O quinto capítulo explica a aplicação desenvolvida para testar o SoF, o Mosquito.Net e os resultados obtidos. As conclusões são discutidas durante o capítulo 6.

2 Frameworks

Um framework é uma estrutura conceitual básica utilizada para resolver problemas complexos. Consiste de um programa que pode ser facilmente estendido e reusado [Johnson 2000]. Um framework indica como um problema deve ser decomposto e no caso de programação orientada a objetos é representado por um conjunto de classes abstratas e interfaces e também pelo relacionamento entre suas classes [Roberts 1998]. Frameworks são abstrações, e representam a generalização da solução de vários problemas reais.

Frameworks são peças fundamentais da moderna engenharia de software. São geradores de aplicações que são relacionadas a um domínio específico [Lucena et al. 2001].

Um framework é composto por diversos padrões de projeto simples e geralmente faz uso da inversão de controle, que é, ao invés de apenas a aplicação invocar uma API (*Application Programming Interface*) e receber o retorno da execução, a aplicação implementa interfaces ou estende classes abstratas do framework, que são invocadas no momento adequado [Johnson 2000].

A utilização de frameworks é vantajosa, pois estes são ferramentas úteis para o reuso, não apenas de código, mas também de análise e projeto [Roberts 1998]. Também proporcionam uma simplificação do código desenvolvido para as aplicações, aceleram o desenvolvimento e facilitam a manutenção.

As desvantagens são que o desenvolvimento sobre frameworks demanda tempo de aprendizado dos desenvolvedores e, em diversos casos, tornam a aplicações maiores e poderosas que o necessário e conseqüentemente com a adição de mais camadas de código a aplicação resultante se torna mais lenta [Johnson 2000].

O usuário, cliente, de um framework é o programador ou desenvolvedor que constrói as aplicações. Para desenvolver uma aplicação utilizando um framework é geralmente necessária a implementação de subclasses e a configuração do relacionamento dos objetos.

Na próxima seção será abordado o processo de desenvolvimento dos frameworks, observando quais os fatores relevantes a serem observados durante este processo. Já na seção 2.2, os frameworks serão classificados. A conclusão do capítulo será apresentada no item 2.3.

2.1 Processo de desenvolvimento de frameworks

O processo de desenvolvimento de frameworks deve partir da generalização de casos concretos [Roberts 1998], a partir deles observam-se as abstrações. Estas generalizações podem surgir de vários aspectos:

- Dividir e quebrar problemas grandes em menores a fim de encontrar componentes similares;
- Encontrar aspectos que apesar dos nomes diferentes nas aplicações originais, atuam de maneira similar no sistema;
- Utilizar parametrização e configuração para eliminar diferenças.

A estrutura interna dos frameworks tem que ser desenvolvida para minimizar o acoplamento entre ele e a aplicação [Baumer 1997], porém os frameworks devem permitir que os desenvolvedores de aplicação tenham flexibilidade no desenvolvimento [Pree et al. 1997].

De acordo com Pree [Pree et al. 1997], os frameworks devem consistir de áreas estáticas ou "congeladas" ("*frozen spots*") e de áreas flexíveis ou "quentes" ("*hot spots*"). As primeiras áreas, as estáticas, definem a arquitetura geral do framework, ou seja, os módulos básicos, os componentes e os relacionamentos entre eles. Estas partes são imutáveis em qualquer instanciação do framework de aplicações. Em contraste, os *hot spots* representam as partes adaptáveis do framework, isto é, as partes onde os programadores, os clientes, podem adicionar o próprio código e funcionalidades específicas ao seus próprios projetos.

Quando se desenvolve uma aplicação concreta sobre um framework, os *hot spots* são especializados e adaptados de acordo com as necessidades e requisitos específicos do sistema em desenvolvimento.

Wolfgang Pree ainda defende que é indicado que os desenvolvedores interessados em construir frameworks devem estudar vários frameworks já existentes para tomar como base. Um problema sobre essa abordagem é que a documentação interna dos frameworks não é geralmente disponibilizada ou é muito fraca. Outro aspecto é que a tarefa de explorar frameworks apenas para estudo pode ser bastante tediosa [Pree et al. 1997].

Sobre o caminho a percorrer para desenvolver um framework, Ralph Johnson e Don Roberts da University of Illinois [Roberts 1998][Johnson 2000] indicam as seguintes estratégias:

2.1.1 Desenvolvimento orientado a exemplos

- O primeiro passo é analisar o domínio do problema e coletar exemplos de programas a serem desenvolvidos ou integrados com o framework.

- O número de exemplos deve ser entre quatro e cinco. Este é um número ideal, mas na realidade, pode ser difícil encontrar ou ter disponível esse número de aplicações para análise.

- A partir destes exemplos projeta-se uma abstração que cobre todas as aplicações. Esse projeto é feito observando as similaridades entre os programas tomados como exemplos e buscando representar cada uma dessas similaridades apenas uma vez na abstração sendo projetada.

- É importante observar que o design deve ser baseado em padrões de projeto, tanto por serem soluções já verificadas, mas também, por serem mais conhecidos pelos desenvolvedores que venham a desenvolver sobre o framework.

- Por fim o framework é testado integrando os exemplos de programas com o framework gerando aplicações sobre o mesmo, onde cada exemplo é uma aplicação separada.

2.1.2 Desenvolvimento iterativo

- Inicia-se com o design inicial do framework e a prototipação de algumas pequenas aplicações.

- A partir delas e do framework, constrói-se uma aplicação real, concreta.

- Em seguida refatoração deve ser efetuada sobre o framework e a aplicação tentando separar o que deve fazer parte de um ou de outro. Essa refatoração se dá, pois com o framework e aplicação em estado inicial, se torna difícil indicar quais os trechos podem ser genéricos (*frozen spots*) e quais são específicos (*hot spots*).

- Estende-se tanto o framework, quanto as aplicações e a partir deste ponto o processo se repete sempre utilizando refatoração e estendendo o framework e as aplicações.

2.2 Classificação de frameworks

Em adição aos processos sugeridos, os autores Ralph Johnson e Don Roberts [Johnson 2000][Roberts 1998] dividem os frameworks em duas classificações, ‘caixa-branca’ e ‘caixa-preta’, as características de cada um destes tipos serão listadas abaixo:

2.2.1 Frameworks ‘Caixa-Branca’

- São customizados através da implementação de subclasses;
- Ênfase em herança;
- O programador cliente precisa conhecer a estrutura interna dos frameworks;
- São mais simples e mais fáceis de projetar;
- São mais difíceis de aprender e requerem mais programação para integração com a aplicação.

2.2.2 Frameworks ‘Caixa-Preta’

- São customizados através de configuração;

- Ênfase em polimorfismo;
- O programador cliente precisa conhecer as interfaces do framework;
- São mais complexos e difíceis de projetar;
- São mais fáceis de aprender e requerem menos programação para a integração com a aplicação.

2.3 Conclusões

O ponto de partida deste trabalho de graduação foi o estudo dos frameworks, analisando o conceito deste tipo de software. Pela sua peculiaridade e características especiais, como por exemplo, possibilitar o reuso, não só de código, mas também de análise e projeto, os frameworks possuem na literatura um *guideline* de como deve ser o seu processo de desenvolvimento.

A classificação dos frameworks também é relevante para análise posterior do framework produzido por este trabalho.

3 Aplicações Sociais

Aplicações sociais ou softwares sociais podem ser definidos como aplicações que permitem, estendem ou agregam valor ao comportamento social humano [Coates 2003]. Isto quer dizer que softwares sociais são um subconjunto dos softwares que estão relacionados ao aumento das capacidades sociais e colaborativas do ser humano. Esta ampliação de possibilidades sociais é permitida pela exploração de benefícios providos pelos computadores e principalmente pela internet, como por exemplo: a remoção de limitações do mundo real, como dificuldades de comunicação em tempo real, dentre várias outras.

É importante salientar que as aplicações sociais não são substitutas da interação humana pela interação via dispositivos eletrônicos, mas visam a expansão e ampliação das possibilidades dessa interação [Rockwell 1997]. Um exemplo dessa idéia seria uma aplicação de mensagens instantâneas, que não busca substituir a conversa presencial, mas visa permitir que as conversas ocorram, mesmo quando os participantes não estão todos no mesmo local, na mesma hora.

A idéia de ampliar as possibilidades do comportamento humano através de dispositivos eletrônicos não é nova. Segundo Christopher Allen no artigo: "*Tracing the Evolution of Social Software*", a primeira referência do uso de computadores para colaboração entre usuários data de 1945, próximo ao fim da segunda guerra mundial onde Vannevar Bush concebeu o que ele chamou de '*memex*' [Allen 2004].

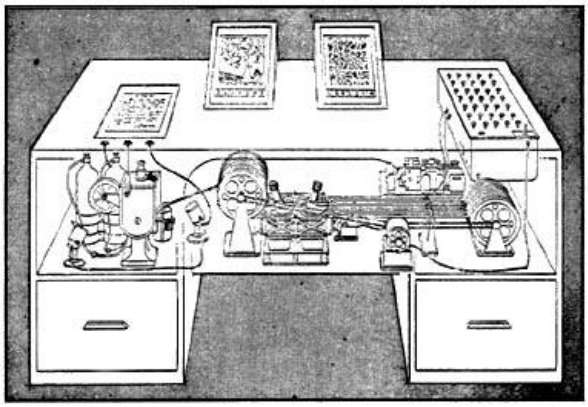

Figura 3.1 – Dispositivo Memex na forma de mesa.

O 'memex' era um dispositivo onde um usuário poderia armazenar seus livros, registros e comunicados e permitia que quem acessasse o dispositivo pudesse buscar conteúdo de forma rápida e flexível. Permitia então a expansão da capacidade de memória e da comunicação do indivíduo [Allen 2004].

Ao longo dos anos essa idéia continuou sendo explorada por novos artefatos e abstrações e também com o surgimento de novos conceitos como "*Groupware*" que é definido como processos intencionalmente relacionados a grupos de pessoas e os

softwares que os suportam e também a idéia de CSCW – "*Computer-Supported Collaborative Work*", ambos os conceitos surgidos na década de 1980 [Allen 2004].

O termo 'Software Social' ou 'Aplicação Social' surgiu no início da década de 1990, mas seu conceito era restrito a utilização de hipertexto de maneira aberta. Mas apesar de vários softwares sociais já estarem em plena execução, apenas em 2002 o termo passou a se popularizar, através da criação do *"Social Software Summit"* por Clay Shirky [Allen 2004] [Shirky et al. 2007].

A popularização dos computadores e também a crescente mobilidade dos dispositivos eletrônicos, que estão cada vez menores e mais poderosos, as aplicações sociais estão num estágio de crescimento elevado.

As redes sem-fio, computadores de bolso, sensores de localização e outras tecnologias permitem as pessoas agirem juntas de maneiras e em situações onde a ação coletiva nunca foi possível antes [Rheingold 2002].

> *"As aplicações do futuro das indústrias de tecnologia da informação e comunicação não serão dispositivos de hardware ou programas em software, mas serão práticas sociais."* [Rheingold 2002]

Este capítulo aborda na próxima seção o estado atual das aplicações sociais listando os tipos mais comuns e populares. Em seguida uma nova abordagem para estas aplicações é apresentada, com a conclusão do capítulo realizada em seguida.

3.1 Aplicações Sociais atuais

Atualmente existem vários tipos de aplicações e softwares sociais, os tipos mais comuns e populares são listados e exemplificados a seguir:

3.1.1 E-mail

O correio eletrônico, que se apresenta como um dos mais simples e antigos softwares sociais, permite que se escreva, envie, armazene e receba mensagens através sistemas de comunicação. Não é em tempo real, o que pode ser observado como vantagem, pois não há necessidade do receptor está online no momento do envio da mensagem.

Também podem ser utilizadas listas de e-mails (*electronic mailing lists*) que permite que ao enviar uma mensagem pra um único endereço, vários receptores a recebam.

3.1.2 *Text chat* / Bate-papo textual

Internet Relay Chat (IRC) e outras tecnologias de bate-papo online permitem que usuários entrem em salas e se comuniquem com várias pessoas ao mesmo tempo. Usuários podem entrar em salas pré-existentes ou criar salas sobre tópicos específicos.

Este tipo de aplicação social expande a capacidade de comunicação dos usuários, permitindo tanto conversas um a um, como de muitos para muitos. O *mIRC* é a aplicação de bate-papo textual que alcançou a maior popularidade.

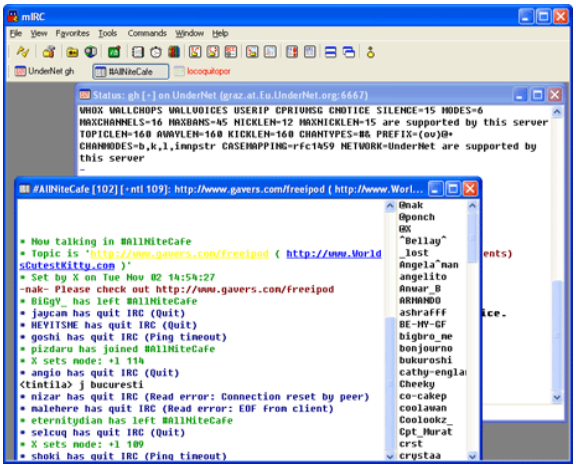

Figura 3.2 – Interface do mIRC

3.1.3 *Instant Messaging* / Mensagens Instantâneas

Um cliente de mensagens instantâneas também permite que uma pessoa se comunique com outra sobre uma rede em tempo real, com privacidade. Uma pessoa só está visível para outra caso a primeira esteja incluída na lista de contatos da última. Caso ambos estejam online, eles se tornam aptos a conversar, também se pode convidar outros contatos para uma conversa. Clientes populares de mensagens instantâneas são: Gtalk, Skype, Meebo, ICQ, Yahoo! Messenger, Windows Live Messenger, Pidgin e AOL Instant Messenger. Alguns destes permitem também conversas utilizando mídias, como imagens, áudio e vídeos em tempo real.

Figura 3.3 - Interfaces Windows Live Messenger e Gtalk

3.1.4 *Internet Forums* / Fóruns

Fóruns permitem que usuários postem um tópico para que outros revisem, respondam ou comentem. As respostas a um tópico são apresentadas geralmente de maneira linear, uma após a outra. Fóruns podem conter várias categorias diferentes e possuem hierarquia de acordo com tópicos e sub-tópicos.

Fóruns várias vezes terminam se confundindo com listas de e-mails (*mailing lists*), como no caso de Google Groups ou Yahoo! Groups, que são classificados como híbridos.

3.1.5 Weblogs

Weblogs, também conhecidos como *Blogs*, são como um jornal online para um indivíduo em particular. O usuário proprietário do *blog*, posta ou escreve no seu *weblog* periodicamente e publica o conteúdo para outros usuários terem acesso a leitura e realização de comentários. Os assuntos abordados nos *blogs* podem ser dos mais diversos, desde o dia-a-dia do proprietário, assim como sua opinião política, resenhas musicais, esportivas ou quaisquer outros assuntos de interesse do mesmo.

Figura 3.4 - Exemplo de Weblog

3.1.6 Mercados virtuais

Mercados virtuais são mercados especulativos criados com o propósito de realizar previsões sobre o mercado real. Ativos são criados e seus valores em dinheiro estão associados a eventos, como por exemplo: a eleição de um novo presidente, ou indicação de novo ministro da fazenda, ou associados a um parâmetro, como por exemplo: o lucro trimestral de uma determinada empresa. É uma maneira de interação mais formal, onde a rede de usuários se forma no interesse de expandir a capacidade preditiva de cada indivíduo do grupo.

Figura 3.5 - Interface Inkling

3.1.7 Serviços de Compartilhamento de Mídias

Os serviços de compartilhamento de média são uma categoria que possui uma grande quantidade de aplicações, como por exemplo, compartilhamento de arquivos em geral (Clientes de Torrent, Compartilhamento *Peer-to-Peer*).

Também é possível compartilhar vídeos e fotos online, enviando o conteúdo para o servidor e disponibilizando para a comunidade em geral. Exemplos são: Metacafe, YouTube, Flickr, Picasa.

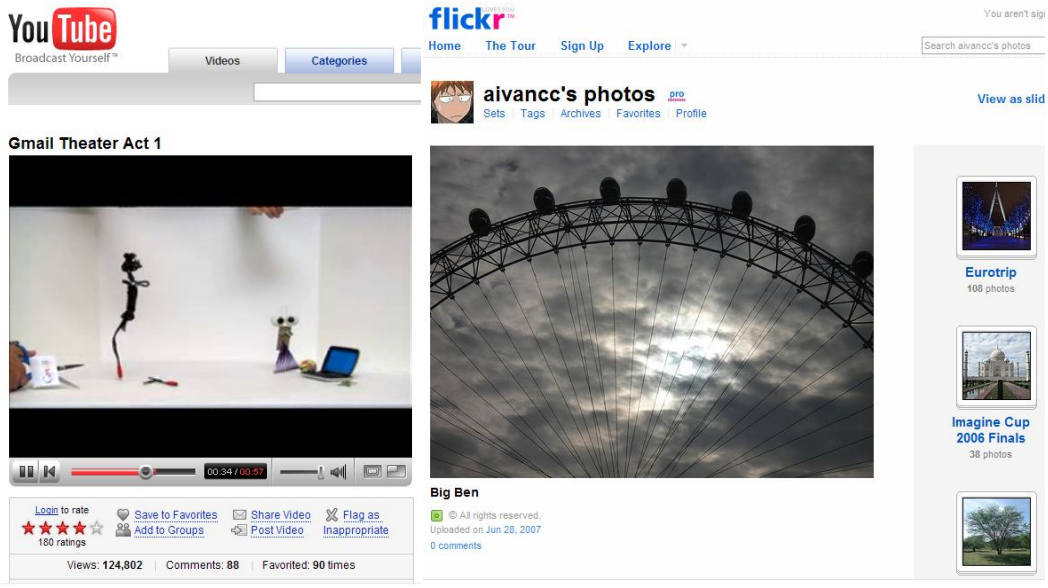

Figura 3.6 - Intefaces do YouTube e do Flickr

3.1.8 Wikis

Um *wiki* é uma página na web na qual o conteúdo pode ser editado pelos visitantes que desejarem colaborar. O foco do conceito é sempre possuir muito conteúdo, e principalmente conteúdo atualizado sem grandes esforços dos criadores do serviço, mas com um pouco de esforço de cada usuário. Exemplos são: Wikipedia; Wiktionary; CommunityWiki; Student Wiki; Wikisource; WikiTravel; del.icio.us.

3.1.9 Editores Colaborativos em Tempo Real

Editores colaborativos em tempo real permitem edições simultâneas para vários participantes em arquivos de texto, planilhas, apresentações e outros documentos. Isto aumenta bastante a capacidade de colaboração dos indivíduos, pois não necessitam estar presentes no mesmo local para gerar um documento em conjunto, nem precisam fazer isso de forma assíncrona, como era costumeiro.

SubEthaEdit e Google Docs & Spreadsheets são exemplos desta categoria de social software.

3.1.10 Serviços de Redes Sociais ou de Relacionamento

Serviços de rede sociais permitem que as pessoas se unam online ao redor de interesses comuns, hobbies, ou outras causas. Por exemplo, alguns *websites* provêem serviços de encontro, onde os usuários postam seus dados pessoais e podem procurar parceiros. Outros serviços permitem *business networking* (Ryze, XING, Ecademy, e LinkedIn), promovem aconselhamento emocional por telefone (Phone Buddies), encontros sociais (Meetup).

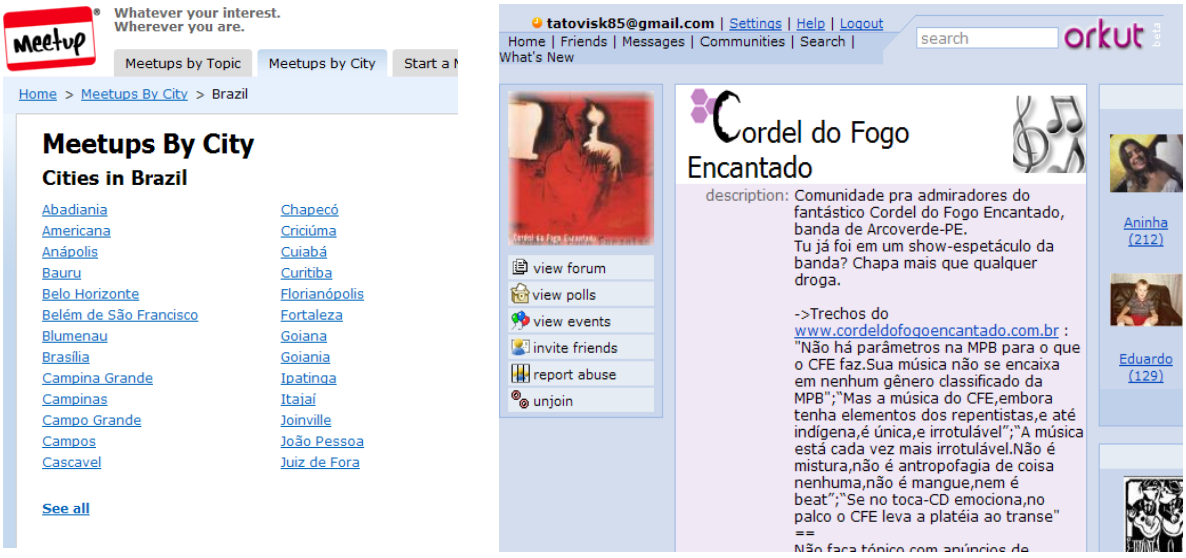

Figura 3.7 - Interfaces do MeetUp e Orkut

3.1.11 Bibliotecas Sociais

Bibliotecas sociais são aplicações que permitem que seus usuários mantenham registro de seus itens de coleção, livros, discos e DVDs. Os usuários podem compartilhar suas coleções e recomendar itens para coleções de outros. Alguns desses serviços oferecem a funcionalidade de empréstimo, onde o usuário mantém o registro dos itens que estão com seus colegas. Exemplos incluem: discogs.com, LibraryThing e lib.rario.us.

3.1.12 Mundos Virtuais

Mundos virtuais são serviços onde é possível encontrar e interagir com outras pessoas num ambiente virtual com elementos do mundo real (realidade virtual). Geralmente, o usuário controla um avatar no mundo e interage com outros através de mensagens de texto ou voz.

- *Massively Multiplayer Online Games (MMOGs)*
  MMOGs são mundos virtuais que possuem vários tipos de pontuação, estágios, níveis, competição, vencedores e perdedores na simulação. MMOGs comerciais (ou, mais adequadamente, massively multiplayer online role-playing games - MMORPGs,) são Everquest e World of Warcraft.

- *Non-game worlds*
  Outros tipos de mundos virtuais não estão relacionados a entretenimento e jogos, pois jogos possuem pontuações, vencedores e perdedores. Ao invés disto, alguns mundos virtuais funcionam como redes sociais de relacionamento como Orkut e Facebook, mas em ambientes 3D. Exemplos incluem: Second Life, ActiveWorlds e The Sims Online.

Figura 3.8 - Interfaces World of Warcraft e Second Life

Estas classificações e definições foram montadas a partir das informações contidas no artigo sobre Social Software na Wikipedia [Wikipedia: Social Software] e também em [Coates 2003] e [Tepper 2003].

3.2 Uma nova abordagem para Aplicações Sociais

As aplicações sociais exemplificadas anteriormente permitem várias maneiras de interação e colaboração entre os seus usuários, expandindo e agregando valor ao comportamento social humano, acelerando processos e mudando, geralmente para melhor, o cotidiano daqueles que as utilizam.

Em contraste com os aspectos positivos das aplicações sociais existentes, observa-se que estas não têm uma preocupação com crises e problemas sociais básicos.

3.2.1 Aplicações Sociais de Larga Abrangência

Esta proposta para aplicações sociais visa à criação de aplicações que envolvam a população na solução dos seus problemas mais básicos como:

- Fome;
- Epidemias;
- Criminalidade;
- Acidentes de trânsito;
- Queimadas;
- Tráfico de entorpecentes;
- Levantamento de recursos.

No aspecto de levantamento de recursos temos exemplos de iniciativas que suas infra-estruturas de tecnologia da informação e comunicação poderiam ser consideradas aplicações sociais de larga abrangência, estas iniciativas são os projetos Criança Esperança [Criança Esperança 2007], que arrecada fundos para projetos da UNESCO [UNESCO 2007] no Brasil e o Teleton [Teleton 2006], que auxilia a AACD [AACD 2007]

3.2.2 Modelo de Negócio

Um ponto relevante para esta nova abordagem para aplicações sociais é que, como estas estão voltadas para o bem da sociedade, seus sucessos estão relacionados a campanhas de marketing para envolver a população e não existe nenhum lucro diretamente associado à comercialização destas aplicações que não envolva o interesse governamental, ou outra instituição interessada em promover uma solução deste tipo por conta de incentivo fiscal ou responsabilidade social.

Observando este ponto, surge o questionamento de quais empresas de software estariam interessadas em partir para o desenvolvimento deste tipo de aplicação. Esta subseção do trabalho de graduação visa apresentar sugestão para obtenção de lucros através do desenvolvimento e implantação deste tipo de aplicação.

A partir do momento que estas aplicações captassem uma boa quantidade de dados e fossem capazes de gerar relatórios e estatísticas consolidadas sobre a sociedade o valor agregado dessas informações poderia interessar a investidores e companhias

externas. O produto oferecido para os clientes estaria diretamente relacionado ao valor das informações obtidas da população.

Um exemplo seria no caso de uma aplicação que mapeasse a criminalidade em uma cidade, identificando áreas de altas e baixas taxas de crime. Poderia ser interessante para uma seguradora de vida obter estas informações para melhor se basear no planejamento dos custos dos seus serviços. Outro exemplo, onde o usuário individual poderia pagar uma pequena taxa, seria no caso de obter via celular ou computador móvel informações sobre o nível médio de tráfego no caminho para uma determinada localidade ou também, quais doenças infecto-contagiosas estão presentes no seu próximo destino de viagem.

Outra fonte de renda para este tipo de aplicação social seria obter uma parcela do lucro de um telefonema ou envio de mensagem de texto no celular juntamente com as operadoras de serviço. Esta já é uma prática comum entre outros provedores de serviço que utilizam ligações.

3.2.3 Discussão

O ponto principal para transformar aplicações que já buscam o monitoramento e combate a problemas sociais em aplicações sociais é prover formas de a população em geral contribuir com a causa, informando dados ao sistema e também obtendo dados e sugestões de ações para reduzir e minimizar os problemas.

3.3 Conclusões

Neste capítulo foram discutidas as aplicações sociais através de conceitos e um breve histórico. Vários tipos de softwares sociais existentes foram listados e foi proposto o desenvolvimento de mais aplicações.

4 SoF – Social Framework

O SoF – Social Framework, é um framework que busca permitir o desenvolvimento mais prático e ágil de aplicações sociais que atinjam uma parcela significativa da população e tragam benefícios para a mesma, por este motivo, o termo ‘larga abrangência’. O framework também pode permitir a aceleração na implementação de outras soluções que não se classifiquem diretamente como aplicações sociais, porém se aproximem na idéia.

Primeiramente serão apresentadas as motivações e objetivos e em seguida as aplicações utilizadas como exemplo para o desenvolvimento do framework e seus requisitos.

4.1 Motivações e Objetivos

A motivação que incentiva a criação deste framework é oferecer aos desenvolvedores de aplicações, empresas de software, organizações governamentais e não governamentais um ponto de início para a implementação de sistemas que visem solucionar ou amenizar problemas sociais que demandem grande ajuda da população.

Ao oferecer várias maneiras de interação com a população e recursos para tratar os dados obtidos o SoF visa permitir uma alta capacidade de absorção dos dados enviados pela população e integrar e processar estes dados de maneira a otimizar a tomada de decisões estratégicas.

O objetivo ao fim do trabalho é ter o SoF como um framework funcional e que atenda os requisitos necessários para ser uma ferramenta de suporte no desenvolvimento das aplicações sociais.

4.2 Aplicações Sociais Exemplo

Os processos de desenvolvimento de framework citados no capítulo 2, o Desenvolvimento Orientado a Exemplos e o Desenvolvimento Interativo indicam que é necessário o levantamento de exemplos base de aplicações que guiem o desenvolvimento do framework.

Para o SoF tomaremos duas aplicações como exemplo e daremos uma visão geral de cada uma, assim como listaremos três macro-requisitos, afim de encontrar algumas similaridades e diferenças entre elas. Deste ponto poderemos levantar os requisitos do SoF.

As aplicações escolhidas foram, primeiramente o Mosquito.Net, que foi proposta para a categoria de Software Design na Imagine Cup 2006 [Imagine Cup 2007], que é uma competição patrocinada pela Microsoft [Microsoft 2007], cujo tema era: "Imagine um mundo onde a tecnologia nos permite viver vidas mais saudáveis". Este projeto foi

classificado em segundo lugar na etapa nacional do Brasil. A segunda aplicação chamada de HelpThem Foi idealizada para auxiliar a organização NineMillion na busca de recursos para a sua atividade fim [Nine Million 2007] .

Aplicações e três macro-requisitos funcionais serão apresentados a seguir:

4.2.1 Mosquito.Net

4.2.1.1 Visão geral

A dengue é um dos principais problemas de saúde pública no mundo. Estima-se que entre 50 a 100 milhões de pessoas se infectem anualmente, em mais de 100 países, de todos os continentes, exceto a Europa [WHO 2007].

O grande problema para combater o mosquito *Aedes aegypti* é que sua reprodução ocorre em qualquer recipiente utilizado para armazenar água. Assim, a prevenção e as medidas de combate exigem a participação e a mobilização de toda a sociedade a partir da adoção de medidas simples, visando à interrupção do ciclo de transmissão e contaminação.

Dentro deste contexto, surge a necessidade de uma solução que atinja a população como um todo, auxiliando a identificação e eliminação dos focos do mosquito da dengue e otimizando os esforços empreendidos no combate à doença.

O Mosquito.Net é uma solução que busca atingir massivamente a população, integrando-a com o governo no combate a dengue, reduzindo gastos e otimizando a ação dos agentes de saúde.

4.2.1.2 Requisitos

[RF01] Obter informações sobre casos de dengue e sua localização

O Mosquito.Net necessita obter dados do máximo de pessoas possível sobre onde ocorrem os casos de dengue e quantos são estes casos. A dengue não está restrita a uma camada social, então é necessário, para um resultado satisfatório, que o Mosquito.Net atinja todas as classes. Isso é possível unindo a abrangência da televisão, aparelhos celulares e internet.

As informações necessárias são: o número de casos de dengue na residência e a localização. De acordo com a interface acessada, essa informação pode vir de diversas maneiras. No caso da Internet, a informação pode já vir completa, com endereço detalhado e informações adicionais, mas para celular e TV Digital, requer muito esforço do usuário, então ele só enviaria o número de casos e o código postal (CEP).

[RF02] Armazenar informações obtidas

É necessário manter informações sobre focos e casos de dengue vindas das diversas interfaces. Para isto devem-se padronizar as informações recebidas, já que são diferentes.

[RF03] Gerar relatórios sobre focos de dengue

Em cima das informações armazenadas é necessária a geração de relatórios que podem visar os mais diversos públicos. Por exemplo, um órgão governamental de saúde deseja saber o quantitativo de casos no geral e quais áreas são mais críticas. O usuário individual quer saber se a área próxima a sua casa ou local de trabalho é uma área de risco. Os agentes de saúde querem saber a que área devem se dirigir e querem monitorar ao longo do tempo o avanço e/ou regressão dos casos.

4.2.2 HelpThem

4.2.2.1 Visão geral

Existem 21 milhões de refugiados ao redor do mundo, dos quais nove milhões são crianças. Essa juventude refugiada não tem educação, lazer e condições de vida adequadas.

O problema dos refugiados é que eles foram forçados a sair das suas terras de origem e tiveram seus direitos violados. Perderam, muitas vezes, a companhia dos familiares e amigos, se vendo obrigados a partir em direção a terras estranhas e desconhecidas, sem recursos para reconstruir suas vidas [Nine Million 2007] .

O HelpThem surge com a idéia de levantar recursos para dar a juventude refugiada chance de aprender e se divertir, promovendo melhores condições básicas de vida e oportunidades de crescimento.

4.2.2.2 Requisitos

[RF01] Obter doações financeiras

O HelpThem busca obter doações do maior número de pessoas possível, então se vale das interfaces da TV Digital, aparelhos celulares, telefone convencional e internet para ter grande abrangência.

As informações necessárias são: a quantia a ser doada e a origem da doação, que para cada caso pode diferir. No caso de telefones convencionais e celulares, é necessário o número do telefone para que possa ser repassado para as operadoras de telefonia. No caso de TV Digital, também pode ser coletado um número de telefone, ou através do código postal, facilitar o preenchimento do endereço completo para envio de boleto. Na internet pode-se obter o número do cartão de crédito ou dados bancários, porém estes dados não devem ser

armazenados e sim os números identificadores das transações indicados pelos bancos e operadoras de cartão de crédito.

[RF02] Armazenar dados das doações

É necessário armazenar os dados sobre as doações para efeitos de cobrança, relatórios e estatísticas. As informações armazenadas devem referir ao método pelo qual a doação foi efetuada e a quantia.

[RF03] Gerar relatórios sobre doações

É necessário saber o total de dinheiro arrecado e a origem deste dinheiro. Também devem ser gerados boletos bancários para envio aos doadores ou informar à companhia telefônica quais de seus usuários doaram.

4.3 Requisitos elicitados para o SoF

Tomando como base os macro-requisitos das duas aplicações exemplo para o SoF – Social Framework podemos levantar os requisitos norteadores do desenvolvimento do framework propriamente dito. Os requisitos do SoF serão divididos em duas categorias tradicionais da engenharia de software: os requisitos funcionais (RF), que são os que especificam funcionalidades, ou seja o comportamento do sistema e os requisitos não-funcionais (NF) que especificam critérios para o julgamento da operação do sistema [Sommerville 2007].

4.3.1 Requisitos Funcionais

[RF01] Permitir diversas maneiras de entradas de dados

O SoF deve permitir que a aplicação a ser construída sobre ele possa obter informações de diversas interfaces, de modo a ampliar a abrangência e o alcance da aplicação. Como os meios de comunicação a serem utilizados serão TV Digital, aparelhos celulares, telefones convencionais e a internet, o SoF deve prover recursos para captura de dados vindos destes dispositivos.

Os tipos de dados recebidos também são diferentes entre as aplicações e o SoF deve estar preparado para tratar esta não-uniformidade.

[RF02] Obter informações complementares

As informações obtidas pelo SoF não serão uniformes, ou seja, dependendo do tipo de aplicação sendo implementado e da origem da informação, os dados recebidos serão diferentes, então cabe o SoF obter informações complementares a fim de uniformizar as informações. Um exemplo concreto é, por exemplo, a interface de TV Digital enviar apenas um código postal (CEP) e a interface web enviar um endereço completo, neste

caso o SoF deve, a partir do código postal montar o endereço com o máximo de precisão possível.

[RF03] Armazenar informações

Com as informações uniformizadas e padronizadas o SoF deve ser capaz de armazenar as informações independentemente da aplicação cliente do framework. O modelo de dados não deve ser especificado pelo SoF, pois é específico de cada aplicação, porém o armazenado deve ser gerenciado pelo SoF através de configuração.

[RF04] Processar informações

As informações recebidas podem necessitar de um processamento após o armazenamento, esse processamento deverá ser customizado de acordo com a necessidade de cada aplicação. Pode-se permitir a integração com serviços de mapas, execução de algoritmos de classificação e mapeamento de regiões, por exemplo.

[RF05] Gerar Relatórios

É de vital importância a possibilidade de extrair os dados processados e montar relatórios a partir deles. Não é a intenção criar uma ferramenta de geração de relatórios complexos, como as já existentes no mercado. No entanto é necessário uma implementação simples que permita uma fácil integração com tais ferramentas.

4.3.2 Requisitos Não-Funcionais

[NF01] Ser escalável

Por necessitar tratar o recebimento de grande quantidade de dados, o projeto do SoF deve permitir sua escalabilidade e fácil extensão. Isto implica que o SoF deve estar preparado para uma futura distribuição onde seus módulos poderão vir a ser executados em máquinas separadas.

[NF02] Ser configurável

Para facilitar o desenvolvimento de aplicações em cima do SoF, este deve ser o mais configurável possível. No estado inicial é aceitável que grande parte da customização parta da implementação de subclasses, mas é fundamental a início da customização via arquivos de configuração.

4.4 Módulos do SoF – *Social Framework*

Tendo como objetivo a implementação dos requisitos do *framework*, um conjunto de componentes, para dar suporte aos desenvolvedores, foi projetado para integrar o SoF. No total, foram seis os módulos identificados: (1) *Data Collector*; (2) *Adapter*; (3) *Data*

Storer; (4) *Processor*; (5) *Reporter*; (6) *Configurator*. O nome dos módulos e a implementação foi escrita em inglês para não limitar o uso do framework aos desenvolvedores com conhecimento na língua portuguesa.

A seguir, estes módulos serão descritos como mais detalhes:

4.4.1 *Data Collector*

Este módulo visa implementar o requisito RF01 permitindo a coleta de dados de diversas maneiras, como por exemplo via *web service*, TCP/IP, *.Net Remoting*, CORBA e SMS. O módulo compreende toda a infra-estrutura de comunicação necessária para a comunicação do SoF com as diversas interfaces.

4.4.2 *Adapter*

É uma biblioteca de classes que visa adaptar as informações recebidas das interfaces através da infra-estrutura de comunicação para uma forma padrão capaz de ser utilizada por todo o restante do sistema. Este módulo é responsável pela implementação do requisito RF02.

4.4.3 *Data Storer*

O Data Storer, como sugerido no nome, é o módulo responsável pelo armazenamento da informação completa, isto é, após passar pelo módulo Adapter, no banco de dados. Este módulo deve ser capaz de armazenar objetos de qualquer tipo, pois depende da aplicação quais as informações serão armazenadas e cabe ao framework gerenciá-las.

4.4.4 *Processor*

O módulo *Processor* coordena a execução do restante do framework, integrando os outros módulos ao redor de si. Além de ser o principal módulo do Social Framework, também implementa o requisito RF04 possibilitando o processamento das informações armazenadas. Este módulo fornece integração com serviços de mapas para melhor visualização das informações armazenadas.

4.4.5 *Reporter*

É um módulo que auxilia a criação e geração de relatórios para as aplicações desenvolvidas sobre o Social Framework. Estes relatórios devem poder ser gerados independentemente dos tipos de dados gerenciados pela aplicação.

4.4.6 *Configurator*

O módulo *Configurator* é responsável por carregar os parâmetros estabelecidos num arquivo de configuração geral do SoF. Para o carregamento dos parâmetros é necessário

leitura de arquivo XML, num formato específico. Este módulo visa atender o requisito não-funcional NF02.

4.5 Implementação

O SoF – Social Framework é composto por sete *namespaces* e sua arquitetura foi projetada para atender aos requisitos identificados através da implementação dos principais módulos descritos na seção anterior.

A tecnologia escolhida para o desenvolvimento foi a plataforma Microsoft .NET, pois provê a infra-estrutura necessária e oferece grande facilidade e praticidade para a implementação na linguagem C#. Foi utilizada a IDE (*Integrated Development Environment*, ou Ambiente Integrado de Desenvolvimento) Microsoft Visual Studio Professional Edition 2005 e a versão do .NET Framework foi a 2.0, compatível com a IDE de desenvolvimento escolhida.

A modelagem da visão de *namespaces* e os diagramas UML foram gerados utilizando o Visual Studio e o Jude Community 5.0.2.

A visão do relacionamento entre os namespaces é exibida a seguir:

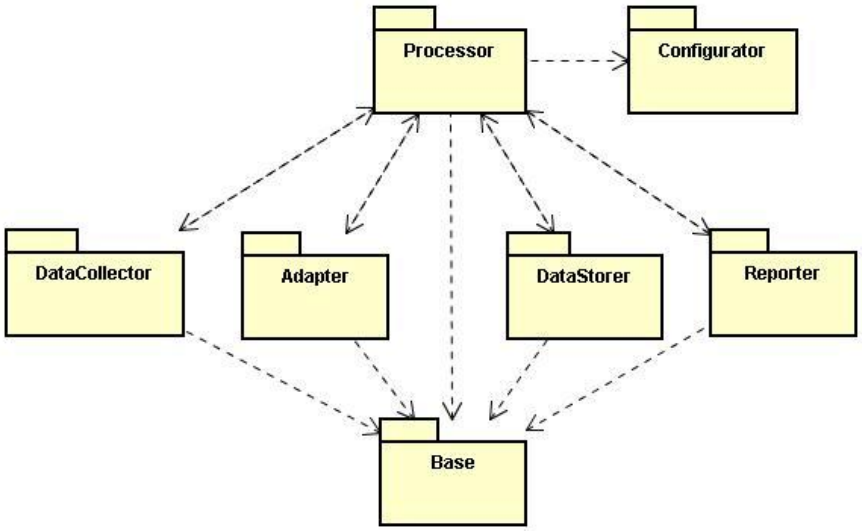

Figura 4.1 – Visão dos Namespaces que compõem a arquitetura do SoF.

Nas próximas seções, serão abordadas as funções de cada um dos namespaces presentes na versão atual do SoF relacionando-os aos módulos que eles pertencem, cada módulo gerou um namespace com o mesmo nome, apenas o namespace *SoF.Base* foi criado para conter classes básicas compartilhadas por todos os módulos. Por fim, será relatado o processo de testes da implementação do SoF.

4.5.1 O Namespace *SoF.Base*

Contém as classe básicas compartilhadas por todo o SoF.

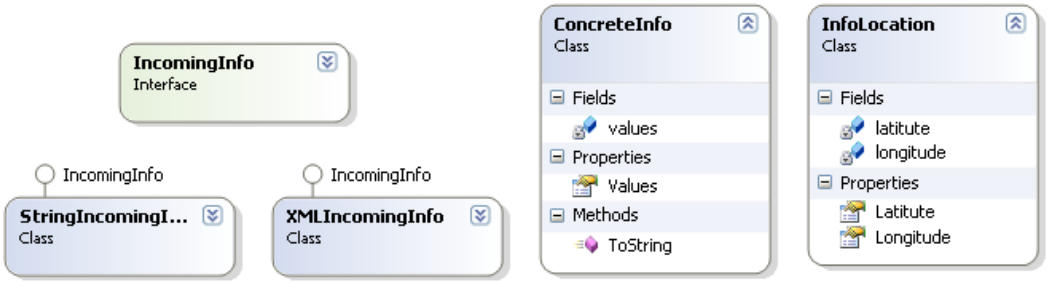

Figura 4.2 - Membros do Namespace Base

IncomingInfo é uma interface que representa os dados vindos dos colaboradores da aplicação social, este dado é tratado e após a padronização, no momento em que está pronto para ser armazenado, deve ter sido convertido para um *ConcreteInfo.*

InfoLocation é utilizado para integração com serviços de mapas, contendo apenas latitude e longitude.

4.5.2 O Namespace *SoF.Configurator*

Criado para carregar as configurações descritas no arquivo de configuração do Social Framework, que recebeu o nome de "*sof-config.xml*".

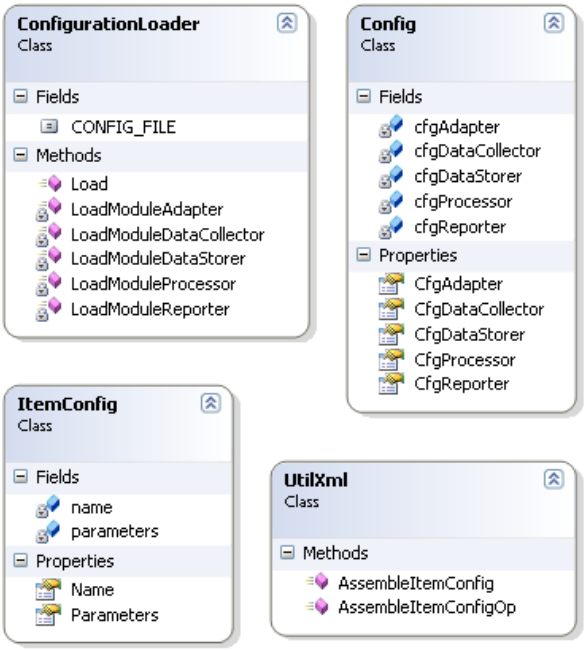

Figura 4.3 - Membros do Namespace Configurator

A escolha do formato XML para o arquivo de configuração foi baseada nas características de extensibilidade, fácil legibilidade para o programador de aplicações

sobre o SoF e também pela fácil integração com a plataforma .NET, que já provê uma API para tratamento de arquivos XML no namespace System.Xml.

Para que o arquivo de configuração possa ser lido, é necessário que ele esteja de acordo com a definição de tipos especificadas no arquivo DTD (*Document Type Definition*) chamado de "*sof-dtd.dtd*". Esta especificação está mostrada em seguida:

```
<!ELEMENT sof (data-collector, adapter, data-storer, processor?,
reporter?)>
<!ELEMENT data-collector (input+)>
<!ELEMENT adapter (template)>
<!ELEMENT data-storer (mapping+)>
<!ELEMENT processor (service*)>
<!ELEMENT reporter (queries)>
<!ELEMENT input (param*)>
<!ELEMENT template (param)>
<!ELEMENT mapping (param+)>
<!ELEMENT service (param*)>
<!ELEMENT query (param+)>
<!ELEMENT param EMPTY>

<!ATTLIST input type CDATA #REQUIRED>
<!ATTLIST mapping table CDATA #REQUIRED>
<!ATTLIST param key CDATA #REQUIRED>
<!ATTLIST param value CDATA #REQUIRED>
<!ATTLIST param op CDATA #IMPLIED>
```

O DTD mostra que há apenas dados contidos nos atributos do arquivo de configuração, pois o último nó, chamado de "*param*" é vazio. Um exemplo de arquivo *sof-config.xml* configurando a execução do Social Framework:

```
<?xml version="1.0" encoding="utf-8" ?>
<sof>
      <!-- Indica os servicos de input a serem executados -->
      <data-collector>
            <input type="tcp">
                  <param key="port" value="2007" />
            </input>
            <input type="webservice">
            </input>
      </data-collector>
      <!-- Indica o Template Method a ser executado -->
      <adapter>
            <template>
                  <param key="class"
                  value="SoF.Adapter.SampleAdapterTemplate" />
            </template>
      </adapter>
      <!-- Indica o mapeamento de dados as operações sobre os dados -->
      <data-storer>
            <mapping table="DengueFocuses">
                  <param key="postcode" value="Location" op="incEql" />
                  <param key="#" value="Reports" op="inc" />
            </mapping>
```

```
            <mapping table="DengueOcurrences">
                  <param key="postcode" value="Location" op="sumEql"/>
                  <param key="numCases" value="Occurrences" op="sum"/>
            </mapping>
      </data-storer>
      <!-- Indica serviços utilizados para processamento das infos -->
      <processor>
            <service>
                  <param key="maps" value="MapPoint" />
            </service>
      </processor>
</sof>
```

No DTD e no exemplo de arquivo de configuração, pode ser observado que existe uma seção específica para cada um dos outros módulos do framework. Vários aspectos do SoF podem ser configurados através deste arquivo.

No exemplo acima, indica-se que o módulo *DataCollector* deve prover serviços de entrada de dados através de conexões TCP na porta 2007 e também utilizar WebService. Para o módulo *Adapter*, a configuração está indicando qual a classe deve ser utilizada para adaptação dos dados recebidos. O mapeamento de dados e operações a serem feitas sobre estes dados estão contidos na seção do *DataStorer*. Para o módulo Processor estão somente sendo indicados quais os serviços devem ser utilizados no processamento dos dados, no caso, um serviço de mapas.

Com a seqüência da descrição dos módulos ficará mais explícito o papel da configuração em cada um deles.

Em relação às responsabilidades de cada membro do namespace *Configurator*, o *ConfigurationLoader* gerencia o preenchimento de uma instância da classe *Config*, que é repassada para o módulo *Processor*. O método *Load*, carrega as configurações contidas no arquivo, utilizando-se de métodos auxiliares específicos para cada módulo. O *parsing* do arquivo XML é feito mais diretamente na classe *UtilXml*.

Um objeto do tipo *Config* contém as configurações de cada módulo armazenadas em mapeamentos de strings para *ItemConfig*, que por sua vez também são mapeamentos de strings em strings. O mapeamentos são feitos utilizando a classe Dictionary do namespace *System.Collections.Generic* da plataforma .NET, que implementa um algoritmo de mapeamento *hash*, criando pares chave-valor. Com esta estruturação é possível armazenar os dados de configuração de forma prática e também acessá-los fácil e rapidamente.

4.5.3 O Namespace *SoF.DataCollector*

Visando possibilitar a entrada de dados no sistema de diversas maneiras foram criadas as classes deste namespace e também foi criado um WebService chamado SoFService.

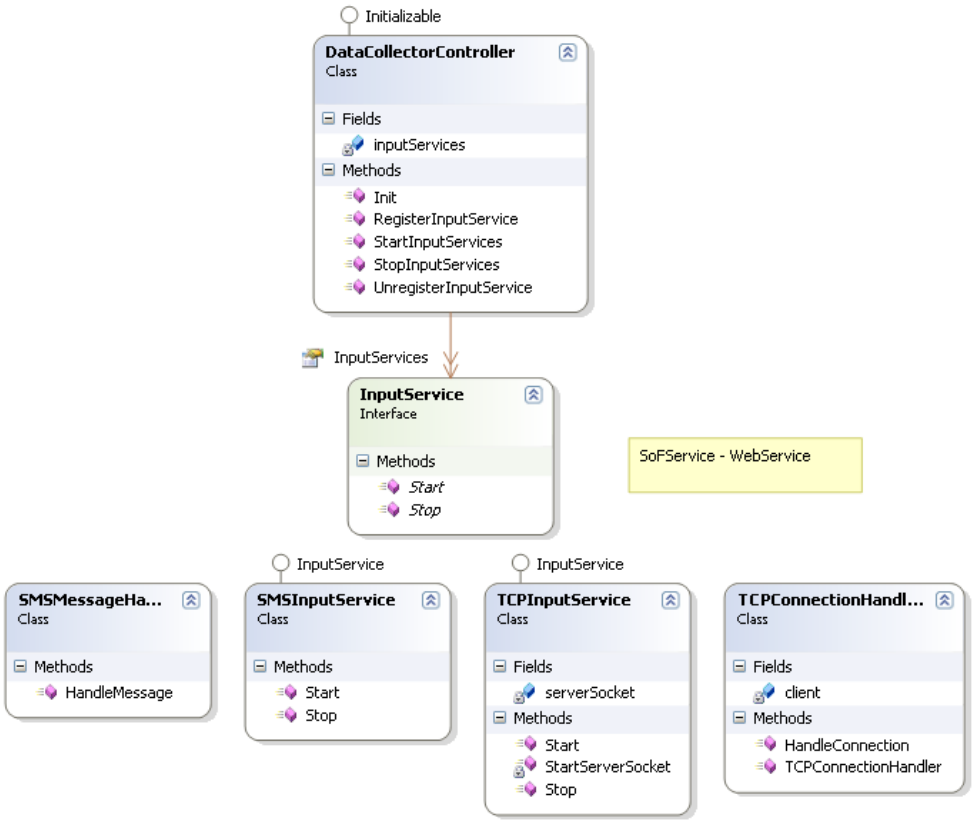

Figura 4.4 - Membros do Namespace DataCollector

A classe *DataCollectorController* é a classe responsável por gerenciar os serviços de comunicação providos pelo SoF. De acordo com a configuração, o *DataCollectorController* registrará na sua lista de serviços, chamada de *InputServices*, todos os serviços de entrada de dados que devem ser executados. O serviços de entradas de dados devem herdar da interface *InputService* que permite o início e parada do funcionamento do serviço através da execução dos métodos *Start* e *Stop*, respectivamente.

Cada serviço de entrada de dados é responsável por tratar a comunicação de forma específica, assim sendo, *TCPInputService*, que implementa a interface *InputService*, juntamente com a classe *TCPConnectionHandler* implementam um servidor baseado na API de Sockets da plataforma .NET provida no namespace *System.Net.Sockets* para receber dados vindo de conexões TCP/IP. De semelhante modo atuam *SMSInputService* e *SMSMessageHandler* em relação as mensagens vindas pelo *Short Message Service*.

As classes de serviço de entrada de dados devem estar cientes do problema da criação excessiva de threads que pode degradar a performance da aplicação. Para solução de tal problema, o SoF utiliza o *ThreadPool* contido no namespace *System.Threading* para enfileirar a requisições e manter o número de threads sempre dentro do aceitável, mantendo o desempenho.

O SoFService é um WebService criado para ser estendido pelo desenvolvedor de aplicações sobre o SoF. Esse serviço também pode ser utilizado para receber dados para a aplicação utilizando o protocolo SOAP que trafega geralmente sobre HTTP.

Para adicionar outros serviços de entradas de dados além dos já providos pelo SoF, basta o desenvolvedor de aplicações implementar a interface *InputService*, registrar o seu serviço no *DataCollectorController* através do método *RegisterInputService* no momento da inicialização do SoF. Com isso podem ser adicionados inúmeros outros serviços, tornando a aplicação apta a receber dados via .Net Remoting, UDP e CORBA, por exemplo.

4.5.4 O Namespace *SoF.Adapter*

Cada serviço de entrada recebe dados de forma diferente dos outros serviços e também, as informações enviadas pelos colaboradores destas aplicações não estão completas para as necessidades das aplicações sociais de larga abrangência. Buscando a padronização e completude das informações foram criados os membros deste namespace.

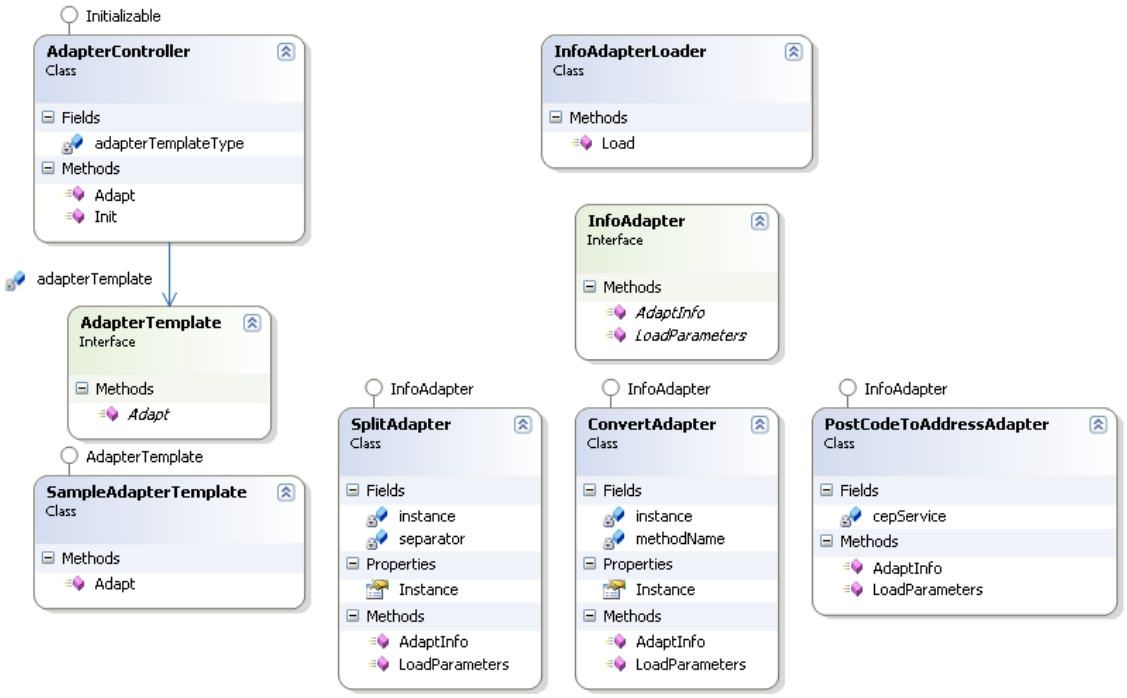

Figura 4.5 – Alguns Membros do Namespace Adapter

Na configuração o desenvolvedor deve indicar qual a classe que implementa a estratégia de adaptação dos dados. Esta classe, que deve ser desenvolvida especificamente para cada aplicação deve implementar a interface *AdapterTemplate*.

O método *Adapt* das classes que implementam *AdapterTemplate* deve ser um *Template Method* [Gamma et al. 2000] definindo o esqueleto do algoritmo que realizará a adaptação utilizando os passos já implementados no framework, ou definidos pelo

próprio desenvolvedor da aplicação. *SampleAdapterTemplate* é um classe implementada com exemplo. O *Template Method* e os passos do algoritmo de adaptação são *hot spots*, pois o desenvolvedor de

O passos para adaptação são implementados pelas classes que herdam de *InfoAdapter*, que é uma interface com dois métodos, *LoadParameters* carrega os parâmetros necessários para execução do passo e *AdaptInfo* executa o passo de adaptação dos dados.

O *AdapterController*, como todos os outros controladores contidos no SoF, gerencia as atividades realizadas pelo seu módulo. Ao ser iniciado, obtém da configuração qual a classe possui o *Template Method* e através do uso da API de reflexão provida pela plataforma .NET no namespace System.Reflection cria uma instância desta classe. Quando o módulo *Processor*, que gerencia todos os outros módulos, invoca o método *Adapt* do controlador, passando a informação coletada pelo *DataCollector*, o controlador invoca o *Template Method* da instância previamente criada. Este método recebe uma *IncomingInfo*, vinda do *DataCollector* e retorna uma instância de *ConcreteInfo*, que é devolvida para o *Processor*. Ambas as classes de informação são do namespace SoF.Base.

O *AdapterTemplate* carrega os serviços de adaptação utilizando o InfoAdapterLoader, requisitando os serviços pelo nome e o carregamento é feito também utilizando reflexão. Esta estratégia adotada para o indicação dos serviços reduz significativamente o código necessário para a implementação do Template Method por parte do desenvolvedor da aplicação sobre o SoF, pois cada serviço pode-se indicar qual o serviço utilizado, passar os parâmetros e invocar a adaptação em apenas uma linha de código.

O SoF provê vários serviços de adaptação de dados. No diagrama são mostrados três destes serviços: *SplitAdapter*, que realiza operações de divisão de string e é útil no parsing de strings recebidas pelos canais de comunicação; *ConvertAdapter*, realiza operações de conversão de tipos, como por exemplo, de string para tipos numéricos, para tipos de data e outros, e *PostCodeToAddressAdapter* que consulta um WebService externo que dado um código postal, retorna o endereço completo relativo a este código, o que é extremamente útil para o tipo de aplicação social que este framework visa auxiliar. Existem outros serviços implementados no SoF e estes também podem ser adicionados pelos desenvolvedores, pois este é um *hot spot* do framework.

4.5.5 O Namespace *SoF.DataStorer*

O objetivo dos membros deste namespace é armazenar as informações que já foram adaptadas pelo módulo *Adapter*. O módulo trata a diferença dos tipos e nomes dos dados e informações a serem armazenados, pois estes diferem de aplicação para aplicação.

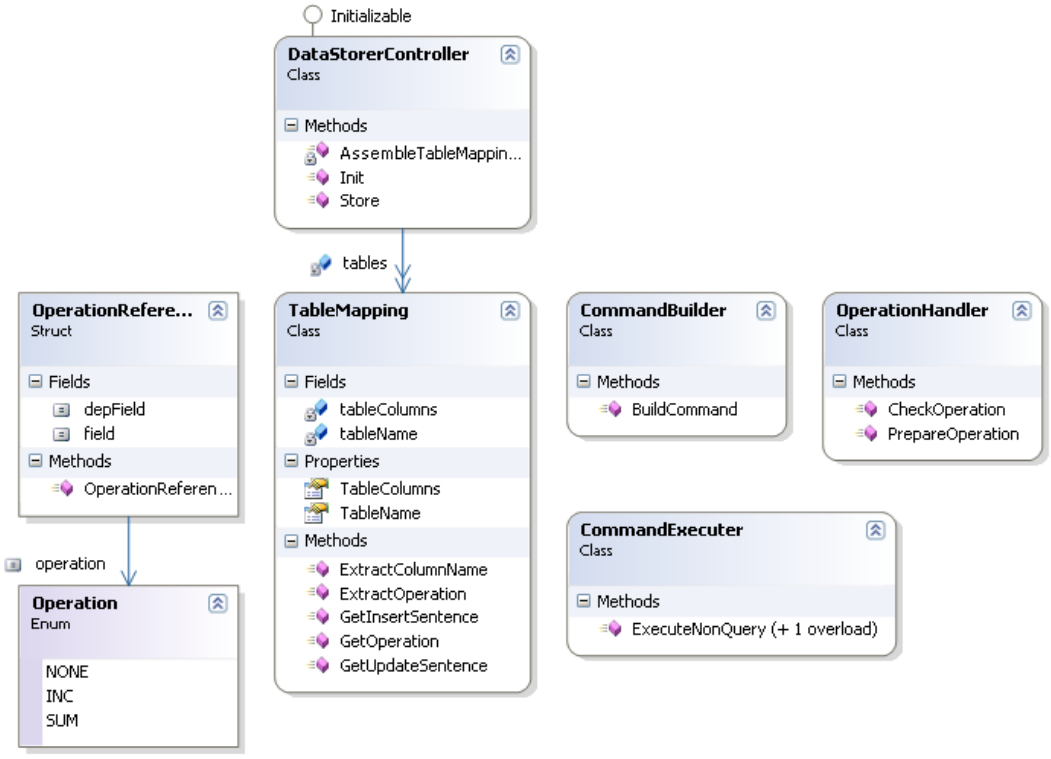

Figura 4.6 - Membros do Namespace DataStorer

Na configuração do módulo *DataStorer* estão contidas as informações de quais informações devem ser armazenadas em que tabelas do banco de dados. E também as operações que devem ser feitas sobre estas informações.

Quando o *DataStorerController* é inicializado ele monta todos os mapeamentos das tabelas do banco de dados com os dados gerenciados pelo SoF. O resultado dessa montagem é uma lista de objetos do tipo *TableMapping*, onde cada objeto corresponde ao mapeamento de uma tabela.

No momento em que o módulo *Processor* repassa as informações a serem armazenadas, o *CommandBuilder*, através do método *BuildCommand*, que é um *Factory Method* [Gamma et al. 2000], monta os comandos relacionando os *TableMapping* com as informações recebidas. Para realizar esta atividade, o *CommandBuilder*, fazendo uso da estrutura *OperationReference*, da enumeração *Operation* e do *OperationHandler*, verifica a necessidade de realizar operações sobre as informações a serem armazenadas, antes do armazenamento.

O desenvolvedor de aplicações sobre o SoF não necessita informar quais os tipos dos dados a serem armazenados, pois isso este dinamicamente identificado e gerenciado pelo SoF no momento da montagem dos comandos.

O SoF suporta atualmente a realização de duas operações sobre os dados, que são incremento e soma. Estas operações funcionam da seguinte maneira: o desenvolvedor especifica no arquivo de configuração, qual campo do banco de dados deve ser

incrementado ou somado e também indica qual o campo do banco de dados a ser tomado como referência para a operação. Para melhor compreensão toma-se o exemplo: necessita-se saber quantos colaboradores da aplicação indicaram um código postal como região crítica, então, para cada vez que a aplicação receber o dado relativo a este código postal, deve incrementar o número de indicações. Neste caso, o SoF antes de inserir a nova indicação de área crítica, verifica se já existe alguma indicação para este código postal, caso sim, incrementa o número de indicações, caso não, insere uma nova. Atualmente apenas é suportada uma operação por tabela.

Após o *CommandBuilder* verificar a necessidade de realização de operação, ele solicita ao *TableMapping* a geração da sentença SQL adequada, que pode ser de inserção ou atualização, que são fornecidas pelos métodos *GetInsertSentence* e *GetUpdateSentence*, respectivamente. Em seguida, os parâmetros são preenchidos na sentença SQL e comando é retornado *DataStorerController*. O comandos são executados pelo *CommandExecuter*.

Toda a interação com o banco de dados faz uso do namespace *System.Data* da plaforma .NET e a conexão com o banco de dados é gerenciada pelo módulo *Processor*, pois é compartilhada com outros módulos.

4.5.6 O Namespace *SoF.Processor*

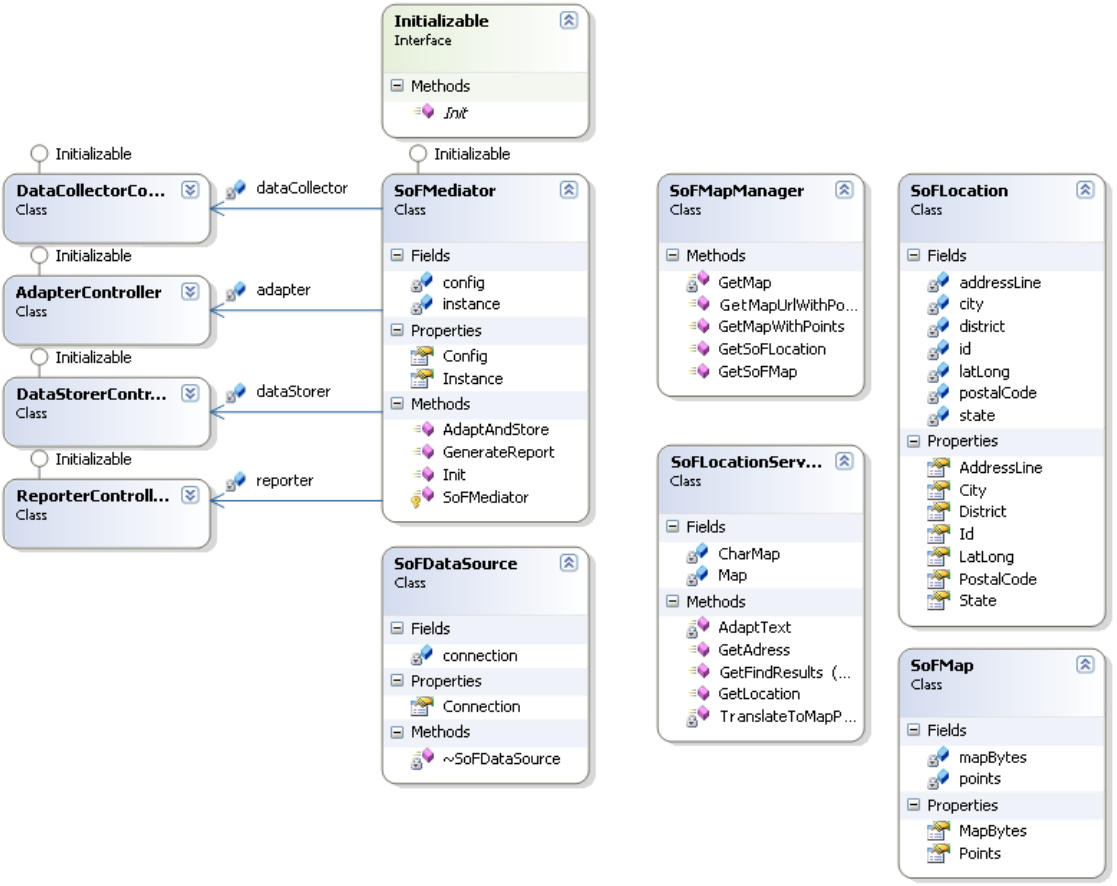

Figura 4.7 - Membros do Namespace Processor e controladores

O namespace *Processor* contém as classes responsáveis pela coordenação do framework, acesso ao banco de dados e também integração com o serviço de mapas. Todos os outros módulos são administrados por este.

A principal classe do Social Framework é o *SoFMediator*, que está associado a todos os controladores dos outros módulos, como exibido no diagrama da Figura 4.7. O *SoFMediator* representa a implementação do padrão de projeto *Mediator* [Gamma et al. 2000], e funcionando iniciando e intermediando a execução das funcionalidades oferecidas pelo SoF. Esta classe também deve possuir apenas um instância em execução, por isto implementa também o padrão Singleton [Gamma et al. 2000].

A interface *Initializable* que é implementada pelos controladores e pelo *SoFMediator* serve para garantir que estes possam carregar as configurações básicas de seus módulos na inicialização do framework.

Como a conexão com a base de dados é utilizada pelo *DataStorer*, pelo *Processor* e também pelo *Reporter*, a responsabilidade de gerenciar as conexões foi atribuída a classe *SoFDataSource*, que repassa uma conexão aberta quando é solicitada e também se encarrega do fechamento dessas conexões. A string de conexão com o base de dados deve ser configurada no arquivo *app.config*, que é o padrão de configuração de todas as aplicações .NET.

As demais classes do namespace, *SoFMapManager*, *SoFLocationService*, *SoFLocation* e *SoFMap* são responsáveis pela integração do SoF com o serviços de mapas MapPoint [MapPoint 2007], provido pela Microsoft [Microsoft 2007].

4.5.7 O Namespace *SoF.Reporter*

O namespace Reporter foi criado para auxiliar o desenvolvedor de aplicações sociais na geração de relatórios.

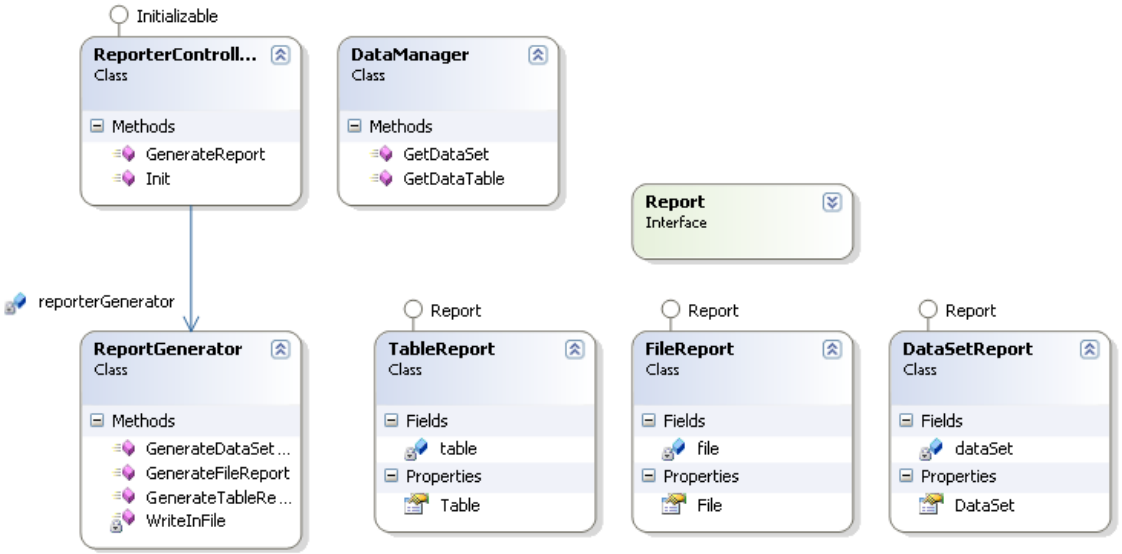

Figura 4.8 - Membros do Namespace Reporter

O módulo *Reporter* pode carregar da configuração algumas consultas determinadas pelo desenvolvedor de aplicações ou também pode receber estas consultas através de passagem de parâmetros em tempo de execução. Dos dois modos, ao receber consultas e a ordem para gerar o relatório, o *ReporterController* através do método *GenerateReport* deve decidir que tipo de relatório deve ser gerado e gerá-lo.

O método *GenerateReport* é um *factory method* capaz de decidir que tipo de relatório gerar através dos parâmetros recebidos. Os relatórios são representados pela interface *Report*, que no SoF, atualmente, é implementada por três classes: *TableReport*, *DataSetReport* e *FileReport*.

As consultas à base de dados são realizadas diretamente pela classe *DataManager*, que obtém a conexão do módulo *Processor*.

4.5.8 Web Services externos

Durante o desenvolvimento do Social Framework foi identificada a necessidade de utilizar serviços disponíveis na internet para prover funcionalidades úteis para as aplicações sociais de larga abrangência. Estes serviços utilizados foram:

4.5.8.1 CEP WebService

O CEP WebService publicado pelo website byJG [byJG 2006] provê serviços sobre a base de dados de códigos postais dos correios, contendo mais de 600.000 logradouros disponíveis para consulta. Este serviço é utilizado pelo módulo *DataAdapter* para obter as informações completas de localização a partir do código postal enviado pelos colaboradores

4.5.8.2 Web Services externos

O Microsoft MapPoint WebService [MapPoint 2007] é um serviço provido pela Microsoft que possibilita a adição funcionalidades baseadas em localização em aplicações desenvolvidas por terceiros. Este serviço é utilizado no Social Framework pelo módulo *Processor* que realiza a integração da aplicação social que utiliza o SoF com o serviço de mapas do MapPoint. Outras funcionalidades do MapPoint ainda podem ser exploradas pelo SoF, como por exemplo os algoritmos de mapeamento de regiões através de polígonos e o planejamento de rotas de deslocamento.

4.5.9 O processo de testes do framework

Antes da implementação de qualquer aplicação sobre o SoF, suas classes foram testadas através de Testes Unitários [Luo et al. 1995]. Para tal, foi utilizado o framework de testes NUnit [NUnit 2006], que é compatível com todas as linguagens da plataforma .NET.

A classes de testes, criadas para este processo, formaram um novo namespace denominado de SoF.Testes. Os testes realizados permitiram a identificação de diversos *bugs* presentes no framework, estes *bugs* foram rapidamente sanados tornando SoF apto para utilização.

4.6 Padrões de Projeto

Esta seção aborda e justifica o uso de padrões de projeto no desenvolvimento do SoF.

4.6.1 Singleton

O padrão de projeto Singleton é um padrão de criação que visa garantir que uma classe tenha uma única instância no sistema, o padrão também provê um ponto de acesso a esta instância [Gamma et al. 2000].

No SoF, este padrão foi utilizado na classe *SoF.Processor.SoFMediator*, pois esta gerencia todo o restante do framework e não faz sentido ter mais de uma instância desta classe em execução.

4.6.2 Factory Method

O padrão de projeto Factory Method também é um padrão de criação, onde há um método responsável por criar um objeto, porém qual subclasse será instanciada ou quais as características do objeto criado vão ser decididos pelo método [Gamma et al. 2000].

O método *BuilCommand* do *SoF.DataStorer.CommandBuilder* é um Factory Method responsável por criar comandos a serem executados no banco de dados. Este método decide que tipo de comando deve ser criado de acordo com o mapeamento das informações concretas nas tabelas. Dependendo das operações a serem feitas nas informações, o comando resultante varia.

Também há um Factory Method na classe *SoF.Reporter.ReporterController*, o *GenerateReport*, que decide quais das subclasses do tipo *Report*, que representa um relatório, deve criar baseado nos parâmetros de entrada recebidos.

4.6.3 Mediator

O padrão de projeto Mediator é um padrão comportamental de objetos que encapsula a maneira como um conjunto de objetos interage. Este encapsulamento provido pelo Mediator tem objetivo de reduzir o acoplamento entre as classes, evitando que os objetos se refiram diretamente uns aos outros explicitamente. [Gamma et al. 2000]

No SoF, este padrão está implementado na classe *SoF.Processor.SoFMediator*, que gerencia como os controladores de cada módulo interagem, evitando que estes se comuniquem diretamente. O SoFMediator, além de atuar na orquestração das ações dos

módulos também atua como intermediário, repassando mensagens de um para outro, reduzindo o número de interconexões.

4.6.4 Template Method

O padrão de projeto Template Method é um padrão comportamental de classes, onde um método define o esqueleto de um algoritmo que realiza uma operação complexa, delegando passos para outras classes. Isto permite a separação da definição do algoritmo da definição de seus passos, que podem ser escolhidos dinamicamente [Gamma et al. 2000]. O Template Method atua definindo as operações primitivas do algoritmo, que outras classes implementam estes passos.

No processo de adaptação dos dados o SoF requisita do desenvolvedor que implemente uma classe contendo um Template Method chamado *Adapt*, e que a classe herde de *SoF.Adapter.AdapterTemplate*. O framework provê vários passos que podem ser utilizados e cabe ao desenvolvedor escolher em que ordem e quais passos serão executados. Um exemplo de implementação deste Template Method se encontra disponível no framework na classe *SoF.Adapter.SampleAdapterTemplate*.

4.7 Conclusões

Neste capítulo foi apresentado o SoF – Social Framework, sua motivação o objetivo de obter um framework funcional, como o que foi construído. A funcionalidade do framework é garantida pelo processo de testes e pela aplicação construída sobre o framework apresentada no próximo capítulo.

Com base nas aplicações exemplos escolhidas para o desenvolvimento deste framework, foram elicitados os requisitos funcionais e não funcionais e a partir destes foram levantados os seis módulos atualmente presentes no SoF: *Data Collector*, *Adapter*, *Data Storer*, *Processor*, *Reporter* e *Configurator*.

Os detalhes de implementação do SoF foram discutidos focando em cada um dos Namespaces gerados a partir das definições dos módulos. O projeto do SoF provê ao framework escalabilidade, pois para aumentar o número de informações que podem ser tratadas pelas aplicações, basta prover uma maior infra-estrutura de hardware, pois o software não limita número de conexões ou impõe qualquer outra restrição no processamento. Ainda para reforçar a escalabilidade, o SoF pode ser distribuído em várias máquinas, conectando suas partes utilizando .NET Remoting ou CORBA.

O processo de desenvolvimento utilizado foi o de desenvolvimento iterativo, utilizando-se das definições das aplicações exemplo e protótipo do Mosquito.Net. A partir daí, gerou-se o framework e realizou-se a integração do protótipo de aplicação existente com o SoF, através de re-design e refatoração.

Quanto à classificação, o SoF está numa posição híbrida entre ser um framework “caixa-branca” ou “caixa-preta”, pois sua customização é feita tanto através de

configuração, quanto por meio de implementação de subclasses utilizando herança. O desenvolvedor de aplicações precisa conhecer as interfaces do SoF, porém ainda necessita ter noção da parte estrutura interna para a integração com aplicação social de larga abrangência sendo desenvolvida.

Os testes unitários das funcionalidades do SoF foram fundamentais na identificação prévia de problemas, e possibilitaram a correção dos erros antes da integração com o Mosquito.Net.

5 *Mosquito.Net*

Como forma de testar a capacidade de aplicação do SoF para a construção de aplicações sociais de larga abrangência, foi integrado ao framework, a aplicação Mosquito.Net.

O Mosquito.Net, como previamente descrito, é uma solução que busca atingir massivamente a população, integrando-a com o governo no combate a dengue, reduzindo gastos e otimizando a ação dos agentes de saúde.

É real problema de obter informações sobre a localização de focos e casos de dengue que afeta a população e afeta também as secretarias de saúde quem sofrem com os altos custos do processo atual do controle da dengue, desperdiçando, algumas vezes, recursos e mão de obra em ações pouco eficazes, localizadas e sem precisão. Para obter maior precisão o Mosquito.Net utiliza três maneiras de entrada de dados: TV Digital, SMS e Website, pois através destes, atinge-se uma grande parcela da população.

Em adição aos problemas levantados, também há a questão da divisão dos recursos destinados ao combate a dengue de forma justa entre estados e municípios, pois tendo maiores informações para tal julgamento, podem-se dividir os recursos de forma a atender a população da melhor maneira possível.

5.1 Visão Técnica do Mosquito.Net

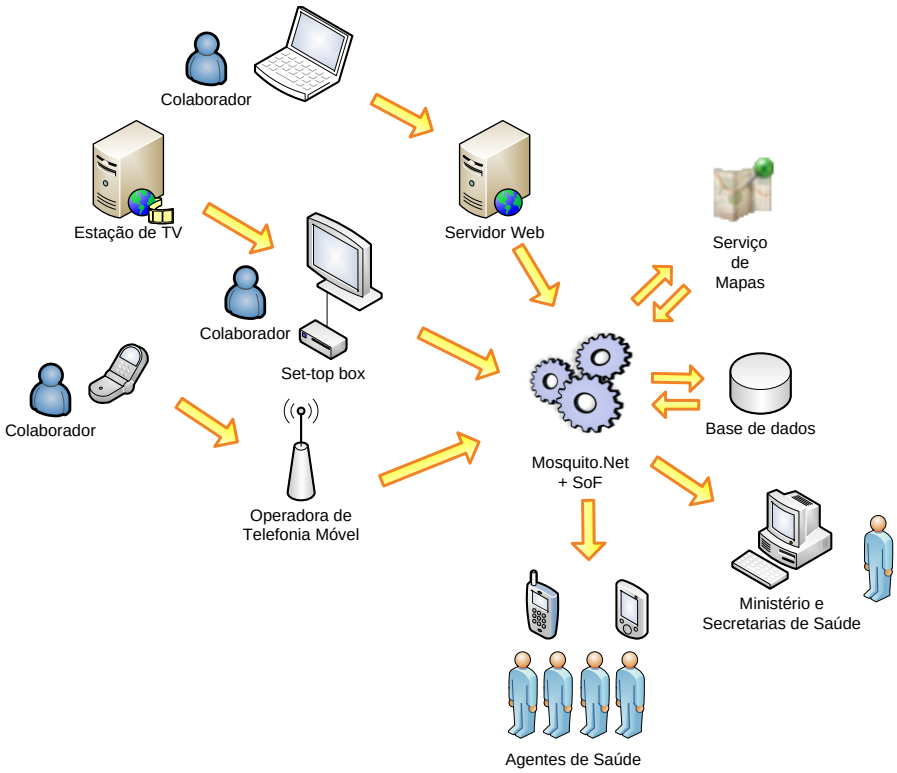

Figura 5.1 - Diagrama de visão geral do Mosquito.Net

No diagrama da Figura 5.1 temos uma visão geral das partes que formam a aplicação social Mosquito.Net. Em seguida, serão abordadas cada uma destas partes separadamente e suas integrações:

No modelo da TV Digital interativa, é possível que a emissora de TV ou provedora de serviço de TVD envie na sua transmissão *broadcast*, além de apenas áudio e vídeo, também aplicações que serão executadas pelos *set-top boxes* dos telespectadores. Caso o padrão de TV Digital terrestre aberta utilizado suporte canal de retorno, o telespectador pode também enviar informações através da TV.

A interface de TV Digital do Mosquito.Net, chamada de Mosquito.TV, foi implementada sobre a plataforma de desenvolvimento do Microsoft Windows Media Center e utiliza como canal de retorno a própria internet para transmitir os dados preenchidos pelos telespectadores para o servidor.

Figura 5.2 - Interface do Mosquito.TV

Em relação às mensagens SMS, estas são enviadas pelo colaborador para um número previamente estabelecido e a operadora de telefonia celular repassa as informações recebidas ao servidor do Mosquito.Net, que trata estas mensagens utilizando o SoF.

Tanto na interface de TV Digital, quanto via SMS, leva tempo para o usuário preencher um longo formulário com informações precisas sobre a localização dos focos e casos de dengue, então, para facilitar a usabilidade, o usuário preenche apenas o código postal, que é composto apenas de números e o SoF obtém as informações complementares para o Mosquito.Net.

No ambiente web, o Mosquito.Net provê um website, chamado de Mosquito.Web que coleta informações enviadas pelos internautas. Utilizando esta interface, o colaborador pode informar dados mais precisos, como número da residência e outras informações que não podem ser obtidas apenas com o código postal.

Após enviadas, todas as informações são tratadas pelo Mosquito.Net, através do Social Framework e armazenadas na base de dados. A partir destas informações o Mosquito.Net mapeia as áreas críticas do país, classificando por criticidade. Os relatórios gerados são disponibilizados na internet, através de uma área restrita no Mosquito.Web, para as autoridades, que neste caso seriam o Ministério e Secretarias de Saúde e também órgãos governamentais ou não, relacionados ao combate da doença.

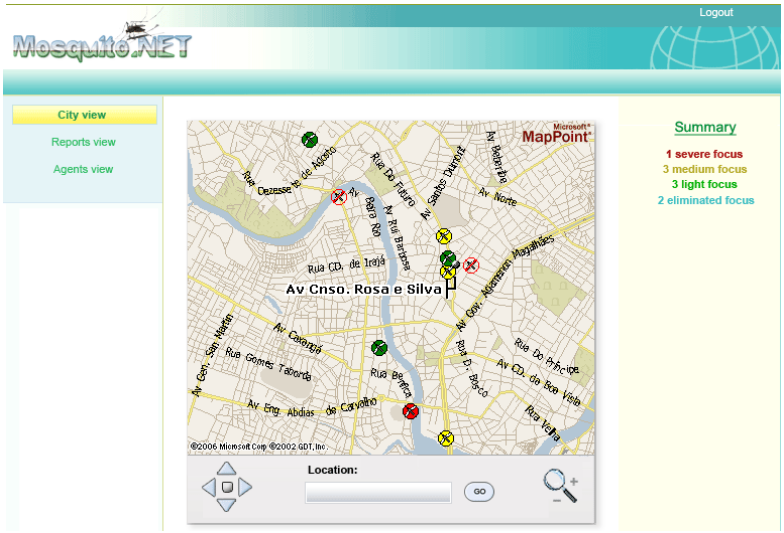

Figura 5.3 - Interface Mosquito.Web Relatório com Mapa

Figura 5.4 - Interface do Mosquito.Web

O Mosquito.Net também provê uma aplicação móvel para auxiliar os agentes de saúde no cotidiano do combate a dengue. Esta aplicação, chamada de Mosquito.Mobile, permite que o Agente de Saúde acesse as informações das regiões críticas e em quais ele deve atuar em determinado momento.

Caso o dispositivo permita, a posição dos agentes de saúde pode ser monitorada pelo Mosquito.Net através de GPS (Global Positioning System), permitindo uma melhor estruturação e planejamento das ações destes agentes.

Figura 5.5 - Interface do Mosquito.Mobile

A geração dos relatórios utilizados tanto no Mosquito.Web quanto no Mosquito.Mobile é auxiliada pelo módulo *Reporter* do *Social Framework*.

5.2 Resultados Obtidos

Este capítulo apresentou o Mosquito.Net, uma aplicação social de larga abrangência para o combate a dengue. Ele foi integrado ao Social Framework, para testar a aplicabilidade deste último na construção de softwares sociais.

O SoF se mostrou capaz de tratar os dados vindos de várias interfaces diferentes, como no caso de TV Digital Interativa, SMS e Web, gerenciando os diferentes dados enviados, reduzindo a participação da aplicação no tratamento destes dados e deixando transparente o armazenamento destes na base de dados. O modo como o Mosquito.Net lida com os dados também foi padronizado, pois era específico para cada uma das interfaces.

A integração com SoF causou a redução aproximadamente 40% no código do Mosquito.Net que não está relacionado a interface, ou seja, código de comunicação e processamento da informações. Este dado reforça a vantagem do uso do Social Framework no desenvolvimento de outras aplicações sociais, pois este provê integração simples para as aplicações além das vantagens de reuso de código e de análise e projeto de software.

Por fim, constata-se que a aplicação do Social Framework no desenvolvimento do Mosquito.Net foi bem sucedida e também auxiliará a implementação de outras aplicações sociais de larga abrangência, pois o framework soluciona os principais problemas deste tipo de aplicação nas áreas de comunicação e gestão de dados.

Apesar dos excelentes resultados obtidos com o Mosquito.Net, apenas a utilização e integração do Social Framework com outras aplicações compatíveis poderá validar sua eficácia e real auxílio para a construção deste tipo de aplicação.

6 Conclusões

Foi identificado que todo o potencial das aplicações sociais não foi explorado pelos desenvolvedores de software e foi vislumbrada a oportunidade de criação de aplicações sociais de larga abrangência que busquem envolver a população na resolução de problema da sociedade.

Estas aplicações possuem características semelhantes, como a obtenção de dados de vários meios diferentes: TV Digital interativa, SMS e internet, armazenamento das informações recebidas, processamento das informações e geração de relatórios.

Baseado nas características semelhantes das aplicações sociais de larga abrangência e buscando facilitar e acelerar os seus desenvolvimentos foi apresentado o Social Framework (SoF), um framework específico para a implementação de aplicações deste domínio.

Os fatores que influenciaram o desenvolvimento de um framework são o reuso de analise, projeto e código provido por este tipo de ferramenta, redução de esforço e simplificação no código das aplicações que se utilizam dele.

O elicitação de requisitos e análise levaram o projeto do SoF a ser composto por 6 módulos, responsáveis por diferentes funcionalidades utilizadas pelas aplicações sociais. Os módulos que formam o SoF são: *DataCollector*, *Adapter*, *DataStorer*, *Processor*, *Reporter* e *Configurator*. Também foi verificado que outros módulos podem vir a ser adicionados no SoF, assim como outras funcionalidades podem providas pelos módulos já existentes, auxiliando ainda mais os desenvolvedores.

Após testes iniciais o SoF foi integrado a aplicação social Mosquito.Net, que visa utilizar a colaboração da população no combate a dengue. Este integração foi realizada de maneira simples, reduzindo o código específico do Mosquito.Net em aproximadamente 40%. Esta métrica foi medida observando apenas o código relativo à comunicação e processamento de informações, pois o SoF, no momento, não visa auxiliar no desenvolvimento dos códigos de interface gráfica, como aplicações de TV digital, móveis e *websites*.

Para implantação desse tipo de aplicações atualmente, o ambiente de TV Digital ainda impõe restrições não encontradas para SMS e o ambiente web. Ainda não existe solução definitiva para o canal de retorno para TV Digital terrestre aberta, o que impossibilita um desenvolvimento definitivo para das aplicações sociais de larga abrangência de com interface de TV Digital.

Existem padrões para TVD que disponibilizam canal de retorno como DVB-MHP (*Digital Video Broadcasting – Multimedia Home Plataform*) na especificação GEM (*Global Executable MHP*) [DVB-MHP 2007], caso o padrão adotado no Brasil possua esta funcionalidade tornar-se-ia viável a implantação do Mosquito.TV, juntamente ao Mosquito.Net.

6.1 Dificuldades encontradas

Durante o desenvolvimento deste projeto, foram encontradas algumas dificuldades que serão relatadas nesta seção.

O ideal para o desenvolvimento de um framework é que se utilize quatro ou cinco aplicações como exemplo para que a análise das similaridades e diferenças e a definição dos *hot* e *frozen spots* se torne mais simples e correta. Porém, no ramo de aplicações que o *Social Framework* visa auxiliar ainda não existem aplicações consolidadas e acessíveis para serem tomadas como base além das tomadas como exemplo durante este trabalho. O SoF também visa proporcionar o desenvolvimento de outras aplicações sociais de larga abrangência.

Durante a implementação da aplicação de TV Digital do Mosquito.Net, o Mosquito.TV, foi necessária a utilização de canal de retorno, que não é suportado pelo simulador de TV digital escolhido inicialmente, o xleTView [xleTView 2005]. Após buscar auxílio com os pesquisadores e desenvolvedores de aplicações para TV Digital do laboratório CENAS do Centro de Estudos e Sistemas Avançados do Recife [CESAR 2007], foi verificada a impossibilidade do uso do canal de retorno através deste simulador, forçando os testes do desenvolvimento a serem feitos em um *set-top box*. Como não há set-top boxes disponíveis no CIn-UFPE foi decidida a utilização do ambiente de desenvolvimento do Microsoft Windows Media Center.

O curto espaço de tempo para o desenvolvimento do Social Framework foi fator redutor para o escopo do projeto. Isto acabou culminando na implementação de apenas uma aplicação sobre o framework. O tempo também influenciou o escopo do SoF.

6.2 Trabalhos futuros

Apesar do SoF já poder ser considerado um framework aplicável para a construção de softwares sociais de larga abrangência, várias atividades ainda podem vir a ser realizadas visando sua expansão, maior validação e também ampliar as possibilidades de uso para os desenvolvedores.

1. **Adição de mais serviços:** a quantidade de serviços atualmente implementados nos módulos do SoF ainda é pequena considerando-se as possibilidades, o aumento da quantidade de serviços é fundamental para aumentar a produtividade dos desenvolvedores de aplicação.

2. **Implementação dos módulos adicionais:** várias aplicações sociais de larga abrangência necessitam controlar as ações para solução do problema que elas combatem, então módulos poderiam ser inseridos no SoF visando atender este requisito.

3. **Melhor Documentação:** A documentação do SoF ainda está precária, impossibilitando que terceiros utilizem o framework sem uma análise do código, então uma boa documentação técnica se faz necessária.

4. **Desenvolvimento de novas aplicações:** Para validar a utilização do SoF no desenvolvimento de aplicações sociais de larga abrangência, é importante testar sua aplicação em outros softwares.

5. **Realizar testes com *set-top boxes*:** Como não foi possível testar a aplicação Mosquito.Net em *set-top boxes* durante o desenvolvimento deste projeto por várias limitações é interessante portar a aplicação para ser compatível com algum disponível.

6. **Versão em Java:** Poderia se utilizar ferramentas de conversão de C#.Net para Java [Sun], já que ambas são similares [Deitel e Deitel 2005], para obter uma versão do SoF em Java permitindo que os desenvolvedores dessa linguagem também tenha acesso ao framework.

7 Referências Bibliográficas

[AACD 2007] ASSOCIAÇÃO DE ASSISTÊNCIA A CRIANÇA DEFICIENTE, 2007. [online] Disponível em: http://www.aacd.org.br/ [Acessado 21 de Agosto de 2007].

[Allen 2004] ALLEN, C., *Tracing the Evolution of Social Software*. [online] Disponível em: http://www.lifewithalacrity.com/2004/10/tracing_the_evo.html [Acessado 06 de Agosto de 2007].

[Baumer 1997] BAUMER, D., 1997. *Framework Development for Large Systems*. Communications of the ACM October 1997/Vol. 40. No. 10.

[byJG 2006] BYJG, 2006. *CEP Web Service*. [online] Disponível em: http://www.byjg.com.br/xmlnuke-php/xmlnuke.php?xml=onlinecep&site=byjg&lang=pt-br [Acessado em 02 de Agosto de 2007].

[CESAR 2007] CENTRO DE ESTUDOS E SISTEMAS AVANÇADOS DO RECIFE, 2007. [online] Disponível em: http://www.cesar.org.br/ [Acessado 21 de Agosto de 2007].

[Coates 2003] COATES, T., 2003. *My Working Definition of Social Software*. [online] Disponível em: http://www.plasticbag.org/archives/2003/05/ [Acessado 05 de Agosto de 2007].

[Criança Esperança 2007] CRIANÇA ESPERANÇA, 2007. [online] Disponível em: http://criancaesperanca.globo.com/ [Acessado 21 de Agosto de 2007].

[Deitel e Deitel 2005] DEITEL, H. E DEITEL, P., 2005. *Java Como Programar*. Sexta edição, Prentice Hall.

[DVB-MHP 2007] MHP, 2000. *Multimedia Home Plataform – The Complete Set of Open Middleware Standards for Interactive TV*. DVB Fact Sheet, July 2007. DVB Project Office.

[Gamma et al. 2000] GAMMA, E., HELM, R., JOHNSON, R. VLISSIDES, J., 2000. *Design Patterns – elements of reusable object-oriented software*. segunda edição, Addison-Wesley Professional.

[IBGE 2005] IBGE – INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA, 2005. *Síntese dos Indicadores Sociais 2004*. IBGE.

[IDG Now! 2007] IDG NOW!, 2007, *Brasil ultrapassa 100 milhões de celulares*. [online] Disponível em: http://idgnow.uol.com.br/telecom/2007/02/15/idgnoticia.2007-02-15.6265530334/ [Acessado 15 de Agosto de 2007].

[Imagine Cup 2007] IMAGINE CUP, 2007. [online] Disponível em: http://imaginecup.com/ [Acessado 03 de Agosto de 2007].

[Johnson 2000] JOHNSON, R. E., 2000. *How to Develop Frameworks*. Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois.

[Lloyd 2003] LLOYD L. S., 2003. *Best Practices for Dengue Prevention and Control in the Americas*, Environmental Health Project, World Health Organization. Strategic Report 7.

[Lucena et al. 2001] LUCENA, C. J. P., MARKIEWICZ, M. E., 2001. *Object Oriented Framework Development*. ACM Crossroads Student Magazine.

[Luo et al. 1995] LUO, G., PROBERT, R. L. E URAL, H., 1995. *Approach to constructing software unit testing tools*. Department of Computer Science, University of Ottawa, Ottawa.

[MapPoint 2007] MICROSOFT MAPPOINT, 2007. *Microsoft MapPoint Web Service*. [online] Disponível em: http://www.microsoft.com/mappoint/default.mspx [Acessado 03 de Agosto de 2007].

[Microsoft 2007] MICROSOFT CORPORATION, 2007. [online] Disponível em: http://www.microsoft.com/ [Acessado em 18 de Agosto de 2007].

[Nine Million 2007] NINE MILLION, 2007. *Facts*. [online] Disponível em: http://www.ninemillion.org/ [Acessado em 04 de Agosto de 2007].

[NUnit 2006] NUNIT.ORG, 2006. *NUnit*. [online] Disponível em: http://www.nunit.org/ [Acessado 14 de Agosto de 2007].

[Pree et al. 1997] PREE, W., SIKORA, H., 1997. *Design Patterns for Object Oriented Software Development*. ICSE 97, ACM.

[Rheingold 2002] RHEINGOLD, H., 2002. *Smart Mobs: The Next Social Revolution Infrastructure for Social Software*. Perseus, Cambridge, Mass.

[Roberts 1998] ROBERTS, D., 1998. *How to Design a Framework*. Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois.

[Rockwell 1997] ROCKWELL, R., 1997. *An Infrastructure for Social Software*. IEEE SPECTRUM, March 1997, IEEE.

[SBTVD 2007] SISTEMA BRASILEIRO DE TV DIGITAL, 2007. *Sistema Brasileiro de TV Digital*, Ministério das Comunicações. [online] Disponível em: http://sbtvd.cpqd.com.br/ [Acessado 15 de Agosto de 2007].

[Shirky et al. 2007] SHIRKY, C., LAWLEY, L., MAYFIELD, R. 2007. *Many 2 Many, a Group Weblog on Social Software*. [online] Disponível em: http://many.corante.com/ [Acessado 05 de Agosto de 2007].

[Sommerville 2007] SOMMERVILLE, I., 2007. *Software Engineering*. oitava edição, Pearson, Addison Wesley.

[Teleton 2006] TELETON, 2006. *AACD / Teleton* [online], Disponível em: http://www.teleton.org.br/ [Acessado 21 de Agosto de 2007].

[Tepper 2003] TEPPER, M., 2003. *The Rise of Social Software*. netWorker Magazine, September 2003, ACM.

[WHO 2007] WORLD HEALTH ORGANIZATION, 2007. *Health Topics: Dengue*. [online] Disponível em: http://www.who.int/topics/dengue/en/ [Acessado 08 de Agosto de 2007].

[Wikipedia: Social Software] *Social Software*. [online] Disponível em: http://en.wikipedia.org/wiki/Social_software [Acessado 06 de Agosto de 2007].

[xleTView 2005] XLETVVIEW, 2005. *xleTView Emulator* [online] Disponível em: http://xletview.sourceforge.net/ [Acessado 08 de Agosto de 2007].

Data e assinaturas

22 de agosto de 2007

Renato Viana Ferreira
(Aluno)

Carlos André Guimarães Ferraz
(Orientador)