Sistema Integrado de Controle de Veículos

SICV

Projeto de Arquitetura de Software

Recife, 18 de abril de 2008

APRESENTAÇÃO GERAL DA ARQUITETURA DO SISTEMA:

1 - Alguns dos requisitos não-funcionais que o nosso sistema atende são:

1.1 – Ser um sistema em tempo-real.
1.2 – Ser escalável até um certo limite.
1.3 – Ser disponível somente para as partes interessadas(viaturas, atendentes e despachantes).
1.4 – Fazer com que as informações de entrada sejam extraídas com qualidade e sejam repassadas às viaturas.

2 - Neste primeiro momento estamos levando em conta as seguintes hipóteses:

2.1 – Sempre haverá um atendente, um despachante e uma viatura para atender uma ocorrência.
2.2 – As viaturas não vão estar em movimento.

3 - O que o sistema faz:

Quando um reclamante faz uma ligação solicitando o serviço, a atendente utiliza o seu módulo de funcionalidade para localizar a ocorrência num mapa real e cadastra informações sobre ela no sistema. Após o cadastro da ocorrência, o sistema opera automaticamente procurando algumas viaturas para atender a ocorrência, após isso, ele repassa para um despachante a decisão de chamar uma viatura. Quando a viatura é notificada através do sistema, ela recebe as informações da ocorrência e uma sugestão de rota e se dirige diretamente para o destino. Além disso, nesse momento é aberto um canal de comunicação entre o despachante e a viatura.

4 - Quais são as escolhas de desenvolvimento de software que aumentam a eficiência do sistema:

4.1 – As partes clientes da aplicação precisam fazer o mínimo de operações possível, por isso, ele é implementado numa arquitetura Cliente-Servidor que utiliza a “Web”.

4.2 - O sistema usará mapas do Google Maps para que os dados sejam extraídos com maior exatidão e não ocorra muito atraso de processamento.

4.3 - Haverá um Login/Logout para que somente pessoas cadastradas possam usar o serviço.
4.4 - A velocidade na execução e na comunicação do sistema é caracterizada pelo fato dele ter canais de comunicação de resposta rápida como “chats”.
4.5 – O sistema vai funcionar na Web porque nós estamos usando a API do Google Maps, já que os seus mapas já realizam algumas histórias com pouco esforço.
4.6 – O sistema está modelado em Camadas porque fica mais fácil de dividir as tarefas e de implementá-las.
4.7 – Alguns *Design Patterns* como *Facade e Iterator* e alguns *Architecture Pattens* como *Repository e Layered Systems* são observados na arquitetura do sistema.

5 - Componentes Externos de Interação com o Sistema:

5.1 – O “browser” do atendente no qual são exibidos as telas de uso para o cadastro de ocorrências.
5.2 – O “browser” do despachante no qual são exibidos as telas de uso para o rastreamento das viaturas e a comunicação.
5.3 – O “browser” do computador de bordo da viatura no qual são exibidos as telas de uso para a visualização da rota no mapa e para a comunicação com o despachante.
5.4 – Idealmente, teríamos o GPS que geraria as posições das viaturas e encaminharia os dados através de requisições. Na verdade, ele foi substituído por uma classe de Simulação.
5.5 – O banco de dados presente no servidor local que garante a persistência do sistema.

6 - Interfaces dos Agentes de Usuário com o Sistema:

6.1 – Haverá uma página HTML para o Login no sistema, com uma caixa de texto na qual são inseridos a identidade e a senha, além disso, há um botão para que sejam repassadas as entradas para o Servidor analisar.
6.2 – Haverá um mapa que representará a cidade e servirá para localizar as ocorrências, referenciar as posições dos veículos e indicar as rotas.
6.3 – Haverá um “chat” no qual as pessoas envolvidas numa conversa preencherão os campos de entrada e visualizarão as mensagens enviadas por ambas as partes.
6.4 – Haverá componentes de interface padrão de páginas HTML como formulários e botões.

7 – Funcionalidades Principais:

7.1 – O cliente usará as funcionalidades do mapa para encontrar a posição mais exata possível da ocorrência, entre elas: modificar o zoom do mapa, o tipo de exibição(Terrain, Hybrid, Map), mover a área visível do mapa, receber rotas selecionar viaturas e modificar rotas.

7.2 – Haverá uma descrição suscinta da rota, dizendo por quais ruas se direcionar e as distâncias e tempos aproximados.

7.3 – Relocalizar a posição da ocorrência.

7.4 – Quando ativo a funcionalidade, o sistema vai ler as posições das viaturas e reposicionar no mapa automaticamente.

7.5 – O policial da viatura e o despachante poderão se comunicar e trocar dados entre si.

7.6 – O despachante e a viatura serão acionados pelo sistema automaticamente.

7.7 – Informações de qualquer fluxo de ação das pessoas envolvidas e do sistema será armazenado num “log” e alguns dados são armazenados em banco de dados.

7.8 – O sistema gerará a rota e repassará para a viatura selecionada que poderá visualizar e se orientar para chegar no local o mais rápido possível.

7.9 – Haverá um Login/Logout para controle de sessão.

8 – Situação da Arquitetura na Segunda Etapa de Desenvolvimento:

Como não tínhamos desenvolvido nenhuma arquitetura rigorosa na etapa anterior a característica de Modificabilidade de Arquitetura de Sistemas, em Métodos Ágeis como o XP, não foi realizada na arquitetura.

Também não vamos analisar as possíveis mudanças na Arquitetura do Sistema pois nós observamos que a probabilidade de haver mudanças é muito pequena, já que, as Escolhas de Projeto garantem com boa exatidão os requisitos do sistema.

MODELOS ARQUITETURAIS:

O nosso sistema foi modelado em dois tipos:

1 – Modelo de Camadas: Assim modelado:

Top – Interface
Fachada
Negócio
Repositório de Dados

Down – Entidades

2 – Modelo Cliente-Servidor: Assim modelado:

Servidor Web: Google Server
Servidor Local: SICV Server
Clientes:

a. Attendant

b. ForwardingAgent
c. Vehicle

VISÕES ARQUITETURAIS DO SISTEMA:

1 - Visão de Módulo:

– Diagrama de Classes:

A idéia da nossa divisão em módulos está baseada no fato de que um módulo(classes ou pacotes) tem a sua estrutura lógica bem definida, organizada e auto-completa, ou seja, não depende de nenhum outro módulo para completar o seu sentido ou existir.

Descrição:

O *DiagramaClassesModulos* mostra a estruturação do sistema no Servidor Local e segue o padrão *Facade, Repository* e o Modelo em Camadas, ou seja, é a representação a nível de classes do sistema em camadas utilizando os padrões *Facade e Repository.*
Já no *DiagramaClassesPacotes* é representada a estruturação desse sistema local em pacotes definidos e hierárquicos, no caso de um ser interno a outro. O desenvolvimento dessa visão arquitetural ajudou na definição das histórias 2, 3, 4, 5, 6, 8 e 9. Além disso, essa visão serve na evolução da *Performance, da Modifiability , da Testability, da Usability e da Business Quality* do sistema. É observado dependências como parte/todo, um para muitos e outros.

2 - Visão de Interação:

– Diagrama de Caso de Uso:

Nós utilizamos o diagrama de caso de uso para saber como os componentes do sistema se interagem. Esse diagrama captura o comportamento do sistema da maneira como ele aparece para um usuário externo. Além disso, ele particiona a funcionalidade do sistema

em relações entre componentes. Logo, nós conseguimos obter facilmente as principais funções do sistema.

Descrição:

Os agentes externos começam interagir com o sistema a partir do Login no sistema, na página principal do sistema. Caso o cliente não consiga se conectar ao sistema será exibida uma mensagem de erro. Ao se conectar diretamente no sistema, o cliente será direcionado diretamente para uma página inicial de uso dependendo do perfil de dados inseridos ao "logar". Se o atender "plotar" no mapa o local da ocorrência e enviar os dados do formulário, será enviado ao servidor o formulário e o algoritmo internamente vai escolher o despachante para atender a ocorrência, a partir daí irá abrir uma nova janela para o despachante operar a nova ocorrência. Note que, enquanto não houver ocorrência para o despachante ele fica em estado de espera. Após ser aberto a nova janela, o despachante irá escolher a viatura para cuidar daquela ocorrência e irá mandar os dados da ocorrência. Além disso, a viatura recebe a sugestão de uma rota, e ela pode abrir ou não uma conversa com o despachante, ou vice e versa. Após resolver a ocorrência, o policial da viatura pode finalizar a ocorrência. O desenvolvimento dessa visão arquitetural ajudou na definição das histórias 1, 4, 6, 7 e 10. Além disso, essa visão serve na evolução da *Performance, do Communicating Concepts, da Testability, da Usability e da Business Quality* do sistema.

- Diagrama de Seqüência:

Nós criamos o diagrama de seqüência pois ele descreve uma seqüência de mensagens trocadas entre regras que implementam comportamento no sistema no tempo.Além disso ela mostra o fluxo de controle atravessando vários objetos. Um outro fator é que ele mostra a seqüência de comportamento de um caso de uso.

Descrição:

No caso do *DiagramaSequenciaGeral* ele mostra o fluxo de mensagens e a direção do fluxo de dados num caso de uso do sistema, operando normalmente. As setas são as mensagens entre os *Actors* do sistema, ou seja, os retângulos superiores. Esse diagrama pode ser usado para descobrir um comportamento do sistema que opere em menos tempo que o inicial. No *DiagramaSequenciaDespachanteServidor* usa-se dos mesmos conceitos acima mas ele foca no comportamento do sistema logo após que a atendente passa as informações para o servidor local e ele começa a procurar viaturas próximas da viatura até antes de ser

plotado no mapa as viaturas e os seus tempos de chegada na ocorrência. O desenvolvimento dessa visão arquitetural ajudou na definição das histórias 3, 4, 5 e 6. Além disso, essa visão serve na evolução da *Performance, da Modifiability, do Communicating Concepts, da Testabilit e, da Usability* do sistema.

- Diagrama de Páginas da Web:

Nós criamos uma visão arquitetural sem utilizar nenhum modelo ou padrão, porque não encontramos, mas precisávamos estruturar as páginas HTML do sistema, para saber quais as telas e quais funcionalidades elas teriam. Inclusive, definindo a sua quantidade.

Descrição:

Os clientes terão como porta de entrada no sistema a página HTML de Login, após terem a autorização de usar o sistema, será exibido para cada tipo de cliente um tipo de página(Vehicle Page Start, Forwardin Agent Page Start ou Attendant Page Start). A partir daí, a exibição de novas páginas HTML estarão orientadas a eventos. Quando o atendente enviar os dados da informação ao servidor e o mesmo encontrar um despachante, será aberta uma nova janela(New ForwardingOccurrence Page). Ao despachante escolher a viatura, essa requisição junto com os dados da ocorrência e o servidor repassará as informações para a página inicial do veículo. Após o carregamento das duas últimas páginas os clientes podem se comunicar por mensagens através do “chat” presente em cada página. O desenvolvimento dessa visão arquitetural ajudou na definição das histórias 1, 2, 3, 6, 7 e 10. Além disso, essa visão serve na evolução *da Modifiability, do Communicating Concepts, da Security, da Safety, da Usability e da Business Quality* do sistema.

3 – Visão de Implementação:

- Diagrama de Desenvolvimento:

Nós implementamos o *DiagramaComponentAlocacao* pois nós localizamos fisicamente componentes do sistema com relação ao hardware(memória, local em disco...) para no caso de sistemas

distribuído, nós começarmos a pensar como interagir essas partes do sistema da melhor maneira possível.

Descrição:

O sistema é divido idealmente em cinco componentes de hardware: * Três representam o lado cliente do sistema. Os browsers são os Agentes de Usuário que rodam as aplicações de cada componente(AttendantComponent, VehicleComponent e ForwardinComponent). Cada um dos componentes se comunica com os Servidores através de protocolos HTTP(Servidor Google) e de Acesso Remoto, as aplicações prescrevem dependências com outras partes de outros componentes.
* O servidor de Máquina Local. Nesse servidor está armazenado o código local e que já foi representado pelo diagrama de classe acima. Esse componente se comunica com outros através de protocolos da internet.
* O Google Server. Esse servidor disponibiliza uma API de desenvolvimento e uso de Mapas. Logo, é nele que se encontra armazenado todos os objetos, interfaces de visualização e funcionalidade referente ao Mapa. Além disso, é utilizado em algumas partes do sistema um “Gadget” do “G!talk” que roda também no servidor mas que sua interface usuário é exibida nos componentes “Attendant” e “ForwardinAgent”. Ele somente não se comunica com o servidor de Máquina local.

O desenvolvimento dessa visão arquitetural ajudou na definição das histórias 1, 7, 8 e 9. Além disso, essa visão serve na evolução *da Modifiability, do Communicating Concepts, da Usability e da Business Quality* do sistema.