Sistema Integrado de Controle de Veículos

SICV

2ª Entrega Projeto Engenharia de Software

Recife, 18 de abril de 2008

1- Membros da Equipe:

a. Carlos Henrique Maciel Sobral Timoteo
b. Diego Albuquerque de Araújo
c. Elliackin Messias do Nascimento Figueiredo
d. Rafael Praxedes Gomes

2- Descrição do Sistema: *Sistema Integrado de Controle de Viaturas - SICV*

O Sistema é baseado na aplicação da Tecnologia da Informação na área de Segurança Pública e Infra-estrutura de Serviços. O CIODS (Centro Integrado de Operações de Defesa Social) tem um sistema que centraliza as ocorrências e as repassam para as subunidades interligadas a ele (PM, PC, CBC, PF...). Ele funciona da seguinte maneira: o solicitante liga para Central do CIODS, um atendente recebe a solicitação e repassa para um despachante de uma subunidade e este delega os atuadores para prestar os serviços. O SICV propõe automatizar as atividades de selecionar a viatura mais apropriada para a ocorrência, sugerir a melhor rota, estabelecer prioridades das ocorrências e monitorar o atendimento, ou seja, auxiliando o trabalho do despachante.

O SICV é uma aplicação web composta por três módulos, um acessado pela viatura, outro pelo despachante e o último pelo atendente.

Os atuadores/viaturas se encarregarão de informar a sua posição mediante o acontecimento de um evento disparado pela Central. Cada atuador terá um Sistema de Posicionamento e terá acesso ao Mapa da Região onde ele se encontra. Além disso, ele se comunicará com a Central via Internet para poder requisitar serviços como: gerar relatório policial, enviar mensagem para a central, chamar mais viaturas para a ocorrência, receber informações sobre a ocorrência da central e outros.

O despachante recebe a ocorrência de um atendente, o SICV localiza a melhor viatura para a ocorrência e envia sugestão da melhor rota, para só assim estabelecer um vínculo de monitoramento entre o despachante e a viatura.

O atendente preencherá o formulário da ocorrência com os dados obtidos do solicitante e enviará a ação diretamente para o SICV(Módulo Despachante) da Central.

O SICV será: uma aplicação distribuída para atender uma grande demanda de solicitações garantindo assim a escalabilidade, desenvolvida na plataforma Java por ser portável e ser uma tecnologia conhecida pelos desenvolvedores do sistema e desenvolvido em “Camadas” para garantir maior modularidade. O sistema também terá um sistema de log e controle de concorrência.

3- Histórias de Usuário:

1ª História: Representar o mapa de uma região no sistema.
Prioridade: Alta

2ª História: Localizar uma viatura no mapa.
Prioridade: Alta

3ª História: Determinar a posição de uma ocorrência no mapa.
Prioridade: Alta

4ª História: O sistema irá sugerir as 5 melhores viaturas e suas rotas para atender uma ocorrência.
Prioridade: Media

5ª História: Gerenciar as ocorrências de acordo com as suas prioridades (gravíssima, grave, média, baixa, baixíssima).
Prioridade: Alta

6ª História: Acionar a viatura escolhida pelo atendente.
Prioridade: Alta

7ª Historia: Permitir a comunicação do policial da viatura com o despachante. O policial passará informações sobre o andamento da ocorrência e receberá um acompanhamento do despachante.
Prioridade: Media

8ª História: Armazenar todas as ações de todos os envolvidos na ocorrência para auxiliar futuras investigações.
Prioridade: Baixa

9ª História: Todas as informações relevantes (ocorrências, pessoas envolvidas, entre outras) devem ser armazenadas em uma base de dados.
Prioridade: Media

10ª História: Mostrar um mapa gráfico na tela do despachante para que ele possa acompanhar a viatura em movimento.
Prioridade: Baixa

4- Riscos do Projeto:

Riscos Imutáveis:

Risco: Não aprender a tempo como implementar a concorrência do sistema.
Probabilidade: Baixa
Severidade: Média
Solução: Restringir a escalabilidade do sistema.
Situação: Como nós não implementamos nada relativo a isso, por conta da probabilidade, ainda é válido esse risco.

Risco: Não aprender a tempo como implementar a persistência do sistema.
Probabilidade: Baixa
Severidade: Alta
Solução: Utilizar arquivos e memória para armazenar os dados.
Situação: Nós implementamos uma boa parte da persistência do sistema e do uso do banco de dados .

Risco: Falta de tempo para se dedicar ao projeto devido a outras disciplinas e a iniciação cientifica.
Probabilidade: Alta
Severidade: Alta
Solução: Administrar melhor o tempo e fazer horas extras.
Situação: Esse risco comprometeu de forma absoluta o desenvolvimento do sistema, já que tivemos uma série de provas, trabalhos para serem entregues, além da Iniciação Científica, na qual tivemos que escrever Relatórios Parciais.

Risco: A ferramenta utilizada para realizar algumas tarefas já estabelecidas, não for capaz de ser auxiliar em novas atividades.
Prioridade: Média
Severidade: Média
Solução: Encontrar novas ferramentas que atendam essa nova necessidade e se possível uma que integre todas as outras ou a maior parte das que já foram implementadas.
Situação: Ainda existe, já que por exemplo o Eclipse não dá suporte à verificação de arquivos JavaScripts.

Risco: O não conhecimento do comportamento e da especificação de partes do sistema e implementá-las de uma maneira inviável.
Probabilidade: Média.
Severidade: Alta
Solução: Estudar e pesquisar em fontes confiáveis para ser ter um conhecimento básico.
Situação: Houve de forma crucial nessa etapa de desenvolvimento do sistema pois redefinimos a idéia e a arquitetura do sistema e assim, tivemos que desprezar código desenvolvido mas que já é implementado no GoogleMaps.

Um outro fator é que tivemos que aprender a usar a tecnologia ligada à nova arquitetura. Além disso, estávamos tentando implementar algumas funcionalidades do sistemas, como “Criar o mapa”, sem ter tempo suficiente para esse fim. Portanto, houve perda de recursos no aprendizado de novas tecnologias (JavaScript, CSS, HTML, Ajax e XML) e na implementação de partes que não vão influenciar no sistema final.

Risco: Haver conflito de horários para programar em pares.
Probabilidade: Alta
Severidade: Baixa
Solução: Sacrificar horários não usuais para programar.
Situação: Devido ao fato de todos os elementos do grupo estar envolvidos com outras atividades também muito importantes como Trabalho e Iniciação Científica, houve diversos conflitos de horário.

Risco: Problemas com gerenciamento das versões do sistema.
Probabilidade: Média.
Severidade: Média
Solução: Utilizar o SVN.
Situação: Ainda existe do mesmo jeito que no início do desenvolvimento, porque existem vários problemas na união das várias versões de código.

Risco: Atraso no Cronograma.
Probabilidade: Alta.
Severidade: Média
Solução: Arcar com as conseqüências e sempre ter em mente que foi feito o melhor possível, para assim nunca abalar a moral da equipe.
Situação: Houve esse risco, por conta de vários fatores:

- Estudar para Provas
- Fazer Trabalhos de Disciplinas
- Um integrante do grupo trabalha e Três fazem Iniciação Científica
- Fazer Relatórios de Iniciação Científica
- Aprender JavaScript, CSS, HTML, Ajax e XML e suas interações.
- Aprender a usar as API’s.
- Muitos erros na construção do código por falta de Experiência.
- Configurar o Tomcat localmente.
- Tempo dispendido para aprender a usar as novas ferramentas.
- Não encontrar a tempo ferramentas de Avaliação de Código(Add-ons)
- Encontrar os livros para escrever a arquitetura.
- Traduzir os livros.
- Aprender a representar os diagramas UML.
- Escrever a Arquitetura e as visões.
- Documentar a arquitetura do sistema.

Todos esses fatores podem ocasionar a não entrega de todas as histórias de usuário.

Riscos Mitigados:

Risco: Usar as ferramentas JUnit, Ant e XPlanner.
Porque: Foi mitigado porque nós aprendemos a usar essas ferramentas.
Como: Tivemos de dedicar algumas horas a manuseá-las, assim desenvolvemos a habilidade de usá-las, de forma intuitiva. No caso do JUnit e Ant, recorremos a tutoriais e sites de ajuda para guiar no aprendizado. Já com o XPlanner, olhando o documentação foi indicado o manuseio direto da ferramenta, já que as suas funcionalidades e características são intuitivas para pessoas que têm certo conhecimento em métodos ágeis.

Risco: Não aprender, a tempo, a usar a tecnologia GPS.
Porque: Seria dispendioso trabalhar com a tecnologia GPS, então muitos recursos seriam absorvidos para pouco retorno. Portanto, nós vamos desenvolver uma classe GPSSimlulator.
Como: Pesquisando em sites, “blogs”, tutorias e olhando sites de empresas fabricantes, foi identificado que cada GPS tem a sua API Empresarial de uso, que usa linguagens específicas. Além do mais, nós não tínhamos recursos suficientes para a compra do equipamento.Por isso, decidimos criar um “simulador de GPS”.

Riscos Adicionais:

Risco: Mudança nas Escolhas de Projeto e na Arquitetura do sistema.
Probabilidade: Alta
Severidade: Alta
Solução: Ver se é necessário essa mudança de escolhas com base na análise de vários fatores como o avanço que vai proporcionar, a facilidade de implementar e os recursos disponíveis que essa mudança pode absorver.

Risco: Implementar páginas HTML que usem tecnologia CSS, JavaScript, Ajax e XML.
Probabilidade: Alta
Severidade: Alta
Solução: Recorrer a tutorias, livros e sites na web para aprender as tecnologias e ajudar no desenvolvimento.

Risco: Não conseguir usar a API do GoogleMaps como ferramenta acessória do sistema.
Probabilidade: Alta
Severidade: Alta
Solução: Olhar o Tutorial do GoogleMaps API, conceitos básicos e principais da API e Google Maps API Reference para identificar e compreender os métodos e atributos usados pelo código.

Risco: O Servidor do GoogleMaps falhar na construção de partes do sistema.
Probabilidade: Baixa
Severidade: Alta
Solução: Não foi pensada nenhum plano para atender a isso já que a probabilidade do servidor do Google cair é muito baixa.

Risco: O Servidor Local não funcionar adequadamente.
Probabilidade: Baixa
Severidade: Alta
Solução: Será preciso realizar a manutenção do sistema.

Risco: Não aprender a usar os diagramas UML
Probabilidade: Média
Severidade: Média
Solução: Tentar representar as interações e os componentes da melhor forma possível, incluindo mais texto informal para explicação.

Risco: Não conseguir montar a Arquitetura e as Visões eficientemente.
Probabilidade: Média
Severidade: Alta
Solução: Dedicar maior tempo à leitura do livro e pesquisar em outras fontes, caso não tiver tempo suficiente fazer o melhor possível.

Risco: Não conseguir documentar a arquitetura do sistema eficientemente.
Probabilidade: Média
Severidade: Alta
Solução: Dedicar maior tempo à leitura do livro e pesquisar em outras fontes, caso não tiver tempo suficiente fazer o melhor possível.

Risco: Não conseguir implementar e usar o banco de dados corretamente.
Probabilidade: Baixa
Severidade: Alta
Solução: Usar outro gerenciador de banco de dados ou gravar a persistência do sistema em arquivos.

5- Implementação e Descrição das Histórias de Usuário

1ª História: Representar o mapa de uma região no sistema.

Descrição: Como essa histórias é uma das mais básicas e principais ela foi implementada primeiro. Após o uso da Google Maps API essa história se restringir a inicializar o mapa do Google Maps e configurar as características iniciais de qualquer mapa, como por exemplo setar o centro do mapa. Os riscos envolvidos são com o servidor do Google.

Duplas Envolvidas: Rafael e Diego.

2ª História: Localizar uma viatura no mapa.

Descrição: A localização da viatura do mapa será feita através de "GMarkers" com o ícone representando a viatura. Isso inclui o fato de que, como as viaturas ise movimentarão, os marcadores irão se movimentar no mapa e representarão as posições ocupadas pelas viaturas num determinado momento. Foi utilizado a comunicação de código Java com JavaScript através de documentos XML. Os riscos envolvidos nessa atividade são relativos ao servidor do Google e ao servidor local que armazenará o arquivo ".xml".

Duplas Envolvidas:

- *Tarefa 1:* Rafael e Carlos – Gerar código de Escrita e Leitura de arquivos XML em Java.
- *Tarefa 2:* Carlos e Elliackin – Implementar os marcadores com os ícones das viaturas.
- Tarefa 3: Elliackin e Diego – Gerar a classe Vehicle em JavaScript e métodos de uso.

3ª História: Determinar a posição de uma ocorrência no mapa.

Descrição: A atendente vai clicar no mapa o local onde ocorreu a ocorrência. Após isso a posição selecionada vai ser passada para o atendente.

Duplas Envolvidas: Carlos e Diego / Diego e Elliackin

- Tarefa 1: Representar a localização da ocorrência no mapa.
- Tarefa 2 : Leitura da posição e envio de dados para o servidor.

4ª História: O sistema irá sugerir a melhor rota para uma viatura que foi acionada.

Descrição: Após o atendente receber a posição de uma ocorrência, o sistema vai sugerir as melhores viaturas e ele vai escolher e encaminhá-la.

Duplas Envolvidas: Elliackin e Diego / Rafael e Elliackin.

- *Tarefa 1:* Sugerir as 5 viaturas, gerar suas rotas e ordená-las.
- Tarefa 2: Mostrar as viaturas sugeridas para o atendente escolher.

Situação: Implementada, mas ainda não conseguimos usar corretamente.

5ª História: Gerenciar as ocorrências de acordo com as suas prioridades (grave, média, baixa).

Descrição: Criar uma classe que implemente essa característica das ocorrências.
Duplas Envolvidas: Carlos e Elliackin.

6ª História: Acionar a viatura escolhida pelo atendente.

Descrição: Ao clicar numa viatura e fazer uma chamada, a viatura vai receber do atendente os dados da ocorrência e vai receber a rota..
Situação: Concluída.
Duplas Envolvidas: Diego e Elliackin.

7ª Historia: Permitir a comunicação do policial da viatura com o despachante. O policial passará informações sobre o andamento da ocorrência e receberá um acompanhamento do despachante.

Descrição: Fazer a comunicação com a viatura escolhida. Foi usado um Gadget do G!Talk para fazer isso.
Duplas Envolvidas: Carlos e Rafael.

8ª História: Armazenar todas as ações de todos os envolvidos na ocorrência para auxiliar futuras investigações.

Descrição: A cada comunicação vai ser armazenado num arquivo ou banco de dados as ações dos envolvidos como uma cópia dos métodos executados.
Duplas Envolvidas: Diego e Elliackin.
Situação: História parcialmente não implementada, pois já criamos o banco resta fazer as tabelas e os métodos de interação.

9ª História: Todas as informações relevantes (ocorrência, pessoas envolvidas, entre outras) devem ser armazenadas em uma base de dados.

Descrição: Os objetos que representam entidades reais do sistema estarão armazenados um banco de dados, com as seguintes tabelas: Ocorrência e Employee(Attendant, ForwardinAgent e do Policial da Viatura).
Duplas Envolvidas: Diego e Elliackin

10ª História: Mostrar um mapa gráfico na tela do despachante para que ele possa acompanhar a viatura em movimento.

Descrição: O despachante terá na sua página “html” um GoogleMap em que ele acompanhará e visualizará as viaturas no mapa.
Duplas Envolvidas: Carlos e Rafael – Rafael e Diego

Dificuldades Encontradas:

A primeira grande dificuldade encontrada foi referente à 10ª história, então, analizando ela pensamos em usar o “Google Maps” e sua API para o desenvolvimento do sistema, pesquisando sobre essa ferramenta verificamos que tinhamos que aprender a usar linguagens como JavaScript, XML e HTML. Então, dedicamos uma grande parte do nosso tempo para aprender a usar essas linguagens, como também a aprender a usar a API do Google e entender a sua estrutura.

Feito isso, começamos a analisar a Arquitetura do sistema, para ver em qual linguagem uma determinada classe deveria ser especificada e quais métodos deveriam existir. Além disso, procuramos definir como seriam as interações entre as diversas partes do sistema e olhando a posição física do código do sistema, vimos que ele seria divido num servidor local com um banco de dados e no servidor do Google.

Prosseguindo, tentamos modelar como seriam as páginas HTML que os Agentes Externos iriam usar definindo sua aparência e os componentes presentes em cada (Formulários, Mapas, Chat...). Tendo feito isso, começamos a reimplementar códigos que estavam escritos em Java mas que deveriam ser escritos em JavaScript. Paralelamente a isso, pesquisamos sobre outras partes do sistema como o comportamento Assíncrono de Interação com o Servidor (Ajax) e como implementar Sessões nas páginas(Login e Logout). Alguns integrantes se dedicaram a produção de algoritmo de geração e leitura de arquivos de extensão “.xml” em Java e em JavaScript, respectivamente.

Portanto, tivemos muita dificuldade em todos as partes de desenvolvimento, por conta da novidade na implementação para os programadores. E por enquanto, não tivemos uma boa produtividade, fator que será incrementado nas futuras etapas de desenvolvimento do sistema, já que os fatores problemáticos levantados no último risco da seção “Riscos Imutáveis” irão diminuir consideravelmente.

Diante disso, nós implementamos de 30% a 40% das histórias de usuários, ou seja, aproximadamente 8 histórias das 17 histórias previstas no total do sistema.

Outra dificuldade encontrada foi encontrar as formas UML corretas que representariam os dados na construção dos diagramas, então fizemos o melhor possível.

Implementação do Banco de Dados:

Nós implementamos o Banco de Dados no Microsoft Acess 2000, existem tabelas representando *Employee*, *Vehicle* e *Occurrence.* Essas tabelas são lidas e atualizadas pelas classes de variação *RepositórioRDB*. A necessidade da criação do banco foi para satisfazer a tarefa 9. Tivemos certa dificuldade, na implementação porque nenhum de nós tinha o conhecimento completo de uso de bancos. Por isso, nos ativemos de implementar primeiramente os seus testes.

Tarefas Adicionais já realizadas:

1. Uso de JPA para implementar a persistência.
    a. Diego e Elliackin
2. Uso de Sevlets para a comunicação das páginas HTML com a Fachada.
    a. Carlos e Rafael
3. Uso do padrão Commands para executar as ações das páginas HTML.
    a. Carlos e Diego
4. Finalização da parte lógica(funcionalidades) das páginas HTML.
    a. Carlos, Rafael e Diego.
5. Uso do Ant para a compilação do Sistema.
    a. Diego

OBS.: A Arquitetura do Sistema se encontra no documento "Projeto da Arquitetura.pdf".