IMPLEMENTANDO UM SIMULADOR DE TRÁFEGO URBANO PARA UMA INTERSEÇÃO COM SEMÁFOROS

Glleddson Fryttys Menezes Leite *
glleddson@yahoo.com

Antônio César Baleeiro Alves (Orientador) *
baleeiro@eee.ufg.br

*Departamento de Computação – Universidade Católica de Goiás - UCG
Av. Universitária n. 1440 - Setor Universitário
74.605-010 Goiânia - GO - Brasil

1 *

Abstract

This article describes the main elements used in the construction of a software for simulation of intersections controlled by traffic lights in an urban traffic corridor. The simulator acts like an intersection in the middle of two road urban protected by using traffic lights. Starting from this application is possible to visualize, analyze and determine the time's fit of a traffic light. Topics related with urban traffic control are approached and also needful aspects for computational model development, including the RUP (Rational Unified Process) and software engineering concepts. In order to obtaining the time's fit of traffic lights are used adjustment performance indicators.

KEYWORDS: Urban Traffic, Traffic Light, Performance Indicators, Simulation, UML, RUP, Java.

2 *

Resumo

Este artigo descreve os principais elementos utilizados na construção de um software para simulação de interseções semaforizadas em uma corredor de tráfego urbano. O simulador representa uma interseção entre duas vias urbanas protegidas pelo uso de semáforos. A partir desta aplicação é possível visualizar, analisar e determinar o ajuste dos tempos de um semáforo. São abordados tópicos relacionados ao controle de tráfego urbano e também aspectos necessários ao desenvolvimento do modelo computacional, que incluem o RUP (Rational Unified Process) e conceitos de engenharia de software. Para a obtenção do ajuste dos tempos dos semáforos são utilizados indicadores de performance.

PALAVRAS-CHAVE: Tráfego Urbano, Semáforo, Indicadores de Performance, Simulação, UML, RUP, Java

3 INTRODUÇÃO

O controle do tráfego de veículos nas grandes cidades, especialmente nos horários de rush, é um dos mais sérios problemas urbanos enfrentados na atualidade. Nas grandes cidades, cerca de 50% dos tempos de viagens e 30% do consumo de gasolina são gastos com os veículos parados nos cruzamentos, esperando que o sinal passe do período vermelho para o verde (DENATRAN, 1979). Restrições orçamentárias e até mesmo físicas, impedem a expansão do sistema de tráfego. Essas restrições levam a um maior aperfeiçoamento no uso de interseções existentes. Com isso, faz-se necessário o conhecimento das grandezas envolvidas no planejamento de interseções, principalmente as semaforizadas, para o aperfeiçoamento do fluxo de tráfico nas mesmas (Iravani and Aash-

tiani, 1999). Dentre essas grandezas podemos citar o atraso médio sofrido pelos veículos e o tamanho da fila média formada durante o período de vermelho do semáforo. No simulador desenvolvido neste trabalho essas e outras grandezas, retratadas em (Júnior, 2002), serão calculadas e exibidas no tempo da simulação.

Os simuladores mais conhecidos são desenvolvidos em outros países e nem sempre atendem as necessidades do nosso perfil de tráfego. Isso faz com que os diversos projetistas de tráfego façam uso de diversos simuladores, aproveitando características isoladas em cada um deles. O desenvolvimento de uma ferramenta dessa natureza, voltada para as condições e características existentes, é de grande importância para o planejamento de interseções semaforizadas.

O simulador desenvolvido por este trabalho consiste em uma ferramenta onde o usuário informa as características da via onde se encontra a interseção, e então visualiza o comportamento dos veículos, realizando assim uma análise a partir dos resultados fornecidos. Esses resultados baseia-se nas equações demonstradas no trabalho de (Júnior, 2002).

Esta aplicação foi desenvolvida utilizando-se a linguagem Java e o RUP*Rational Unified Process*. Os artefatos da modelagem foram produzidos utilizando a UML (*Unified Modeling Language*).

4 ELEMENTOS DO SIMULADOR

Muitas vezes o cálculo de determinada grandeza se torna bastante complexo, por outro lado a observação direta do sistema se torna complicada demais devido as suas variações, nestes casos fazemos uso da simulação. Com a simulação geramos uma "história artificial"do sistema, e a observação dessa "história artificial"para esboçar as conclusões relativas às características operacionais do sistema real que é representado (Banks, 1998). No sentido deste trabalho, o simulador é uma ferramenta para a obtenção de resultados a partir dos dados informados, e para a observação do comportamento do sistema em um nível visual.

Os elementos físicos que fazem parte deste simulador são objetos conhecidos por todos. Um veículo, um semáforo, uma fila de veículos que se forma quando o semáforo passa do sinal verde para o vermelho, e muitos outros, fazem parte do nosso dia-a-dia. No entanto, essa visão simplificada esconde a complexidade que existe em um modelo matemático utilizado para coordenar de maneira fiel todos esses elementos. As ações tomadas por um motorista quando um carro a sua frente pára, a aceleração desenvolvida por um veículo em uma via, ou ainda mais, a distribuição de probabilidade que representa a entrada dos veículos na via, são aspectos que devem ser observados. E pela exigência de uma imitação com o mundo real, é que se faz necessário o desenvolvimento de um bom modelo matemático.

O controle de todos os elementos que compõem esse sistema foi realizado de maneira bastante simplificada. No entanto, uma aproximação de alguns modelos existentes, tais como o IDM (*Intelligent Driver Model*) (Treiber and Helbing, 1999), foram utilizados para o controle dos veículos. Também, algumas distribuições de probabilidade já são largamente utilizados para esse tipo de sistema, uma vez que o seu comportamento possui uma boa aproximação. Mais detalhes serão descritos à frente.

4.1 O Semáforo

No controle de tráfego, o semáforo é um dispositivo que controla o fluxo de várias vias conflitantes que se interceptam em um cruzamento. A autorização da passagem é dada, ou não, através dos sinais luminoso verde e vermelho (Júnior, 2002). As grandezas temporais mais importantes dos semáforos são o intervalo, o ciclo e a fase. O intervalo é o período de tempo em que todas as indicações do semáforo permanecem estáticas, o ciclo é a sucessão de indicações no semáforo que se repetem periodicamente, e a fase é uma parte do ciclo dedicada a um conjunto de movimentos que recebem o direito de passagem simultaneamente.

Em cada fase do semáforo é assegurado aos veículos um conjunto de movimentos dentro da via. Esses movimentos poderão ser o direito de passar pela interseção no mesmo sentido, ou a realização de conversão à direita.

Para este simulador, o tempo de cada fase do semáforo é fixo, ou seja, são estabelecidos um ou mais planos de tempos fixos de funcionamento, que, uma vez definido, sempre será o mesmo para cada fase dentro do ciclo. Os quatro semáforos localizados na interseção terão seus tempos sincronizados, dessa forma o tempo de vermelho de uma das vias deve levar em consideração o tempo de verde da via perpendicular. A Figura 1 ilustra um diagrama de fases/estágios em duas fases.

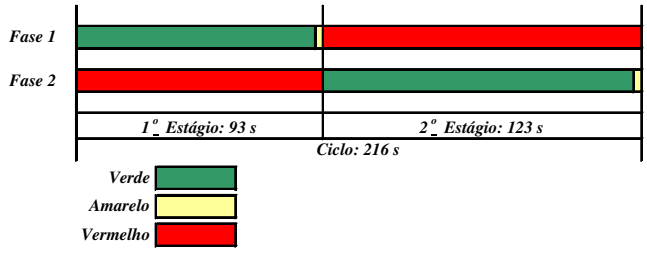

Figura 1: Fase, ciclo e estágios

4.2 Adição de Veículos

Os veículos são adicionados à via, durante a simulação, obedecendo a Distribuição de Poisson (Equação 1) (Jain, 1991). Para inserirmos esses novos veículos fazemos uso de Monte Carlo. Qualquer quantidade maior ou igual a um pode ser desprezada, uma vez que a probabilidade é bem próxima de zero. Então, calculamos a probabilidade de não ocorrência de veículos. Com essa probabilidade pode-se sortear uma

variável aleatória uniforme e determinar se um novo veículo será adicionado (Jain, 1991). A cada segundo, o mecanismo de tempo (*timer*) verifica se um novo veículo será incluído na simulação.

$$P(X) = \frac{\lambda^x e^{-\lambda}}{x!} \quad (1)$$

Onde λ é a taxa de chegada, x é a quantidade de veículos, e é um número irracional, base dos logaritmos naturais e $P(X)$ é a probabilidade de surgir um veículo.

Para este trabalho a probabilidade de um veículo ser incluído na faixa da direita ou na faixa da esquerda é a mesma $(P(X)_{faixa\ esquerda} = P(X)_{faixa\ direita})$. Mas ressaltamos que essa foi uma simples convenção, e que dificilmente representaria a realidade. Para que um modelo mais realístico possa ser elaborado, faz-se necessário que haja procedimentos de coleta em campo, utilizando assim técnicas de coleta de dados específicas, como foi realizado em (Júnior, 2002). Um veículo sendo incluído na faixa da direita, as probabilidades de convergir ou não à direita são iguais $(P(X)_{converte} = P(X)_{não\ converte})$.

A velocidade inicial de um novo veículo será de $20\frac{Km}{h}$. Para um modelo mais realístico, seria necessário elaborar uma tabela de velocidades médias da via que está sendo observada, conforme realizado em (Lees, 2003).

4.3 Variação da Velocidade e Espaço do Veículo

A variação da velocidade de cada veículo foi baseada no IDM (*Intelligent Driver Model*). O IDM assume que cada veículo é uma partícula individual dentro do sistema da via de tráfego. Os parâmetros utilizados pelo o IDM são: desejo de velocidade em uma via livre, desejo de velocidade segura em relação ao veículo à frente, aceleração dentro do tráfego, distância segura em relação ao veículo à frente e aceleração exponencial (Treiber and Helbing, 1999).

Um veículo poderá aumentar a sua velocidade sob duas condições: a sua velocidade não poderá ultrapassar a velocidade máxima permitida para a via (desejo de velocidade) e, o espaço obtido pela alteração da velocidade não poderá ultrapassar a distância de segurança em relação ao veículo da frente (*gap*). Se mesmo mantendo a velocidade o veículo ultrapassar essa distância segura, ele deverá reduzir a sua velocidade até que mantenha uma distância segura com o veículo à frente, ou ainda, ou até que ele pare completamente. Um aspecto importante para o modelo é que, um semáforo impedindo a passagem de um véiculo que está liderando um pelotão é considerado como um veículo parado à frente.

Para este trabalho assumimos que o veículo aumentará ou diminuirá sua velocidade em 1 $\frac{m}{s}$, desde que não ultrapasse a distância segura em relação ao veículo a sua frente. A Tabela 1 mostra como é realizado o cálculo da velocidade e a Figura 2 ilustra o algoritmo para essas atualizações.

Nova Velocidade	Cálculo do Espaço	Condição	Ação
$V_o + U(0,1)$	$S = S_o + Vt$	$S <= S_f + L_f + gap$	Atualiza
$V_o + U(0,1)$	$S = S_o + Vt$	$S > S_f + L_f + gap$	$-1\frac{m}{s}$

Tabela 1: Cálculo da velocidade do veículo

Onde V_o é a velocidade atual do veículo, $U(0,1)$ é uma variável uniforme com valores entre 0 e 1, S é a nova posição do veículo, S_o é a posição atual do veículo, V é a nova velocidade, t é o tempo $(1s)$, S_f é a posição do veículo à frente, L_f é o comprimento do veículo e gap é a distância entre os veículos.

Os cálculos para essas variações são convenções adotadas neste trabalho, para um modelo mais realístico seria necessário explorar como o progresso da distribuição da velocidade dos veículos adapta-se aos modelos de distribuição teóricos (Akçelik and Besley, 2001). Para isso, seria necessário realizar procedimentos de coleta em campo (Júnior, 2002).

```
...
while (vel > velideal){
  vel = vel - (1.0 * Constantes.FATOR_MS);
}
...
vel = velocidadeIdeal(vatual.getVelocidade() + aumentoVelocidade(), velocidade);
espatual = espaco( vatual.getPosicao(), vel);
while (  (espatual > (vfrente.getPosicao() - Constantes.COMP_VEICULO - Constantes.GAP) ) ){
  vel = vel - (1.0 * Constantes.FATOR_MS);
  espatual = espaco( vatual.getPosicao(), vel);
}
...
```

Figura 2: Algoritmo para atualização da velocidade e do espaço

5 INDICADORES DE PERFORMANCE

Os indicadores de performance são as grandezas envolvidas no planejamento das interseções. Uma exigência indispensável para um modelo é definir as medidas de desempenho de tráfico tais como, o atraso médio sofrido pelos veículos e o comprimento da fila média no tempo de sinal vermelho (Akçelik and Besley, 2001).

5.1 Grau de Saturação

Essa grandeza indica o grau de congestionamento de uma via decorrente da oferta do tempo de verde (Júnior, 2002). O grau de saturação é obtido pelas seguintes equações:

$$\gamma = \frac{q}{S} \quad (2)$$

$$x = \frac{\gamma}{\lambda} \quad (3)$$

Onde γ é a taxa de ocupação, q é a taxa de chegada de veículos, S é o fluxo de saturação $\left(\frac{veículos}{segundo}\right)$, x é o grau de saturação e λ é percentual do verde em relação ao ciclo. A

Figura 3 mostra o algoritmo utilizado para cálculo do grau de saturação.

```
...
//Cálculo Ciclo
Ciclo = simCanvas.getTempoVerdeOLLO() + simCanvas.getTempoAmareloOLLO() +
        simCanvas.getTempoVerdeNSSN() + simCanvas.getTempoAmareloNSSN();
...
//Cálculo Verde Efetivo
VE_NSSN = simCanvas.getTempoVerdeNSSN() + simCanvas.getTempoAmareloNSSN() -
          TempoPerdido_Slider.getValue();
...
//Percentual do intervalo verde sobre o ciclo (Lambda)
Lambda_NSSN = VE_NSSN / Ciclo;
...
//Saturação
Saturacao = ((double)Saturacao_Slider.getValue() / 3600.0) * 2.0;
...
//Taxa de Ocupação
TO_NS = simCanvas.getFluxoNS() / Saturacao;
...
//Grau de Saturação
GS_NS = TO_NS / Lambda_NSSN;
...
```

Figura 3: Algoritmo cálculo do grau de saturação

5.2 Atraso Médio por Veículo

Chegando ao semáforo, os veículos param, passam direto, ou apenas reduzem a velocidade. Com isso, uma parte dos veículos sofre algum atraso. O atraso médio por veículo é uma importante grandeza para estimar o desempenho de um semáforo. A relação de Webster é utilizada em inúmeros controles de interseções e ferramentas de análise de tráfego, e é dada pela Equação 4 (McShane, 1990 apud (Júnior, 2002)):

$$d = \frac{C(1-\lambda)^2}{2(1-\lambda x)} + \frac{x^2}{2q(1-x)} - 0.65\left(\frac{C}{q^2}\right)x^{2+5\lambda} \quad (4)$$

Onde C é a duração do ciclo do semáforo, o λ é a relação entre o tempo de verde efetivo e o tempo de ciclo (G_{ef}/C), x é o grau de saturação e q é o fluxo de veículos $\left(\frac{veículos}{segundo}\right)$. A Figura 4 mostra o algoritmo utilizado para cálculo do atraso médio por veículo.

```
...
//Atraso Médio
AM_NS = ((Ciclo*Math.pow(1.0-Lambda_NSSN,2.0))/(2.0*(1.0-Lambda_NSSN*GS_NS))) +
        (Math.pow(GS_NS,2.0)/(2.0*(double)simCanvas.getFluxoNS() * (1.0-GS_NS))) -
        0.65*( Math.pow( (Ciclo / Math.pow((double)simCanvas.getFluxoNS(), 2.0)),
        (1.0/3.0) ) )*Math.pow(GS_NS, (2.0+5.0*Lambda_NSSN));
...
```

Figura 4: Algoritmo cálculo do atraso médio

5.3 Ciclo Ótimo

Essa relação aplica-se a veículos que chegam aleatoriamente a um cruzamento isolado com um semáforo de ciclo fixo. O ciclo ótimo é aquele que fornece o menor atraso. Ciclos muito grandes ou muito pequenos causam grandes atrasos (Júnior, 2002). O ciclo ótimo é dados pela Equação 5:

$$C_0 = \frac{(1.5T_p + 5)}{(1-Y)} \quad (5)$$

Onde T_p é o tempo total perdido pelos veículos e Y é o somatório das taxas de ocupação críticas (γ_{crit}). A Figura 5 mostra o algoritmo utilizado para cálculo do ciclo ótimo.

```
...
//Ciclo Ótimo
YCritNSSN  = Math.max( TO_NS, TO_SN );
CicloOtimo = (1.5 * (double)TempoPerdido_Slider.getValue() + 5.0) /
                                    (1 - ( YCritNSSN + YCritOLLO ));
...
```

Figura 5: Algoritmo cálculo do ciclo ótimo

5.4 Comprimento da Fila Média

O comprimento da fila média pode ser considerado como a quantidade de veículos que chegaram durante o sinal vermelho e formaram uma fila que se dissipará ou diminuirá, assim que houver permissão para passagem (Júnior, 2002).

$$N = \max\left[q\left(\frac{R}{2} + d\right), qR\right] \quad (6)$$

Onde N é a fila média em número de veículos, q é o fluxo de veículos, R é o tempo de vermelho (análogo ao verde efetivo) e d é o atraso médio por veículo. A Figura 6 mostra o algoritmo utilizado para cálculo do comprimento da fila média.

```
...
//Comprimento da Fila Média
VER_NSSN = simCanvas.getTempoVermelhoNSSN() + simCanvas.getTempoAmareloNSSN() -
           TempoPerdido_Slider.getValue();
...
FM_NS = Math.max( simCanvas.getFluxoNS() * ( ( VER_NSSN / 2.0) + AM_NS ),
                  simCanvas.getFluxoNS() * VER_NSSN);
...
```

Figura 6: Algoritmo cálculo do comprimento da fila média

6 PROCESSO DE *SOFTWARE* E O RUP

O processo de software é formado por grupos de atividades que geram resultados observáveis e dessa forma dirigem a produção de um produto de software. Pode abranger o desenvolvimento de um software no seu estado inicial, na sua extensão ou na modificação de um sistema existente (Sommerville, 2003). O RUP (*Rational Unified Process*) é um processo de engenharia de software, que oferece uma direção a ser seguida dentro do processo de desenvolvimento do sistema, com isso oferece uma produção de software de alta-qualidade que atenda as necessidades dos seus *stakeholders* (todos aqueles que interagem de alguma forma com o sistema que está sendo desenvolvido e detém o poder sobre o desenvolvimento do mesmo (Santander and de Vasconcelos, 2003)), dentro de prazos e orçamentos pré-definidos (Kruchten, 2000).

O RUP divide o desenvolvimento em quatro fases denominadas Concepção (*Inception*), Elaboração (*Elaboration*), Construção (*Construction*) e Transição (*Transition*). Cada uma dessas fases é concluída por um marco principal. Essas

quatro fases constituem um ciclo de desenvolvimento e tem como resultado a geração de um software (Kruchten, 2000). Para que um incremento de software seja finalizado nem sempre percorremos todo o ciclo proposto pelo RUP, ou até mesmo, nem sempre iniciamos um novo incremento pela primeira fase proposta.

Os aspectos fundamentais para utilização do RUP neste trabalho é que ele pode ser utilizado como um Processo *Framework*, ou seja, pode ser adaptado e estendido para as necessidades específicas de cada negócio. Na próxima seção descreveremos os fluxos de trabalhos (*workflows*) utilizados nas fases abordadas para o projeto do simulador, e focaremos nas adaptações realizadas para o mesmo.

7 O PROJETO DO SIMULADOR

Neste trabalho serão abordados os fluxos de requisitos, análise e projeto e implementação. Estes fluxos guiarão as atividades dentro das fases de concepção, elaboração e construção. Aqui estaremos interessados em simplesmente descrever as etapas necessárias para se chegar na versão executável do simulador de tráfego.

7.1 Fase de Concepção

Durante esta fase procuramos estabelecer os casos de uso do sistema e delimitar o escopo do projeto. Aqui identificamos todos os casos de uso e descrevemos os mais significativos. Essa fase foi realizada com apenas uma iteração e utilizamos os fluxos de requisitos e análise e projeto.

O fluxo de requisitos teve como resultado o diagrama principal de casos de uso. A Figura 7 mostra o diagrama de caso de uso principal do projeto. Nesta fase também aplicamos diversos refinamentos no diagrama de casos de uso principal, com a finalidade de definirmos melhor o escopo e as funcionalidades do sistema.

No fluxo de análise e projeto criamos um modelo de classe que seja o mais simples possível e que mostre apenas os relacionamentos entre as classes. Para isso criamos o diagrama de classes conceitual mostrado na Figura 8 (Leite, 2003).

7.2 Fase de Elaboração

Nesta fase, a arquitetura de software é descrita e todos os casos de uso e atores identificados. A arquitetura guiará a fase de construção, dizendo respeito a aspectos, tais como: a organização do sistema de software, a seleção dos elementos estruturais e suas interfaces, o comportamento, a composição dos elementos estruturais e comportamentais e o estilo da arquitetura que orientará a organização (Kruchten, 2000). Está fase foi dividida em duas iterações.

Na primeira iteração, o objetivo principal do fluxo de requisitos é criar os casos de uso com base nos requisitos ainda

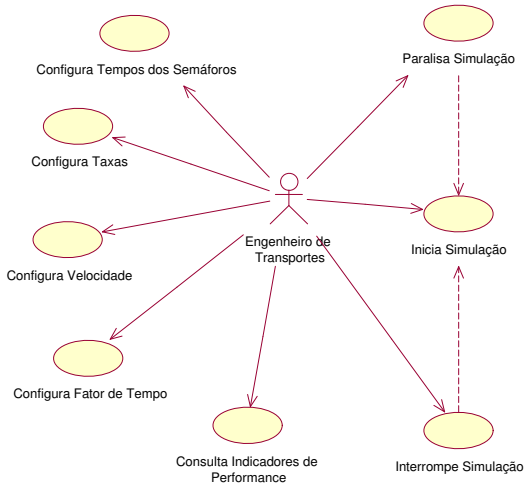

Figura 7: Diagrama de caso de uso principal do projeto

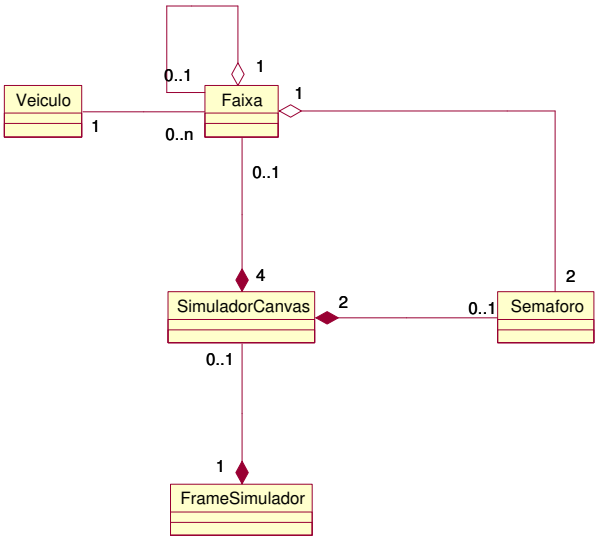

Figura 8: Diagrama de classe principal

existentes. Na segunda iteração, como o restante dos requisitos já foi identificado e esse já alcançou o seu objetivo, nenhum trabalho para esse fluxo será realizado. No fluxo de análise e projeto vamos a partir da análise dos requisitos dos modelos de casos de uso, definir os atributos para as classes. Em um sistema maior poderia haver uma subdivisão das classes entre diversos pacotes, neste projeto não há necessidade dessa subdivisão. Após a criação do diagrama de classes, alguns diagramas que modelam aspectos dinâmicos do sistema podem ser criados (Booch et al., 2000). Foi designado para essa fase o diagrama de gráfico de estados e o diagrama de atividades. A Figura 9 mostra o diagrama de classe final do projeto.

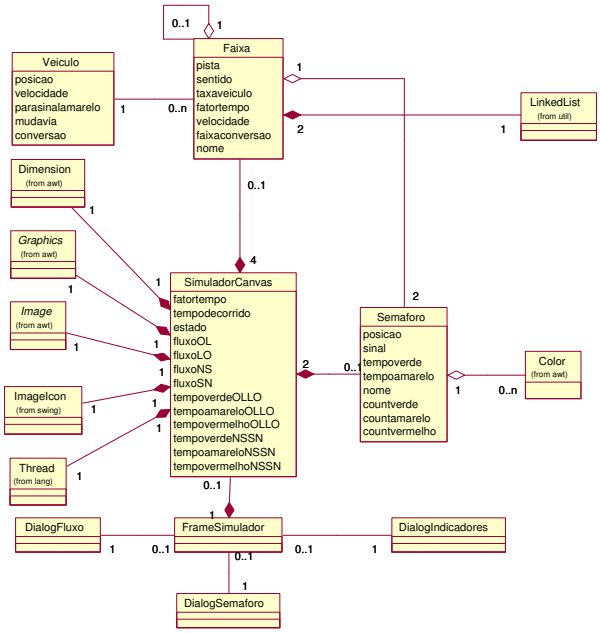

Figura 9: Diagrama de classe final do projeto

7.3 Fase de Construção

Apesar de algumas alterações serem realizadas no fluxo de análise e projeto, os maiores esforços desta fase estão concentrados no fluxo de implementação. Está fase estará centrada em dois fluxos principais: fluxo de projeto e fluxo de implementação.

O fluxo de projeto baseia-se em criar os diagramas de seqüência para os casos de uso mais relevantes e que já foram descritos nas fases anteriores. Esses diagramas transcrevem detalhes da implementação e também o comportamento do sistema em relação às requisições do usuário (Leite, 2003).

7.4 O Fluxo de Implementação

Ao chegarmos neste ponto, todo o sistema já foi compreendido e modelado. O processo de software aplicado neste trabalho nos ajudou a descobrir os detalhes do sistema, a criar uma documentação satisfatória e principalmente a nos conduzir durante todo o processo de desenvolvimento. Apesar de toda a nossa especificação de requisitos ter sido formalizada, foi através de modelos de protótipos que conseguimos enxergar mais facilmente o que seria construído, esse fato é comentado em (Pressman, 2002).

A implementação do simulador foi realizada utilizando a linguagem Java. Essa linguagem foi escolhida por estar bastante difundida no meio acadêmico, por construir aplicativos desktop e para WEB e também por conseguir representar os modelos elaborados ao longo do projeto, uma vez que é uma linguagem orientada a objetos (Deitel and Deitel, 2003).

As principais classes do projeto já foram ilustradas na Figura 9. A classe *SimuladorCavas* é responsável por desenhar todos os elementos do simulador, é essa classe que controla o cronômetro do sistema (*timer*). O tempo dentro do simulador avança de acordo com o fator de tempo informado. Cada veículo que surge em uma das vias é uma instância da classe *Veiculo*, essa classe possui todas as propriedades instantâneas dos veículos, tais como: velocidade, posição, via, faixa e estado. Cada instância da classe *Faixa* possui uma coleção de veículos de uma determinada faixa do simulador, isso se faz necessário devido alguns veículos poderem realizar ou não conversão à direita. As instâncias da classe *Semaforo* possuem suas propriedades principais atualizadas pelo cronômetro do sistema, devido a sua sincronização com os outros objetos desta mesma classe.

A Figura 10 mostra a *interface* principal do simulador. Nesta tela estão os botões de controle do simulador e a entradas para as seguintes informações: velocidade máxima permitida na pista, tempo perdido, fator do tempo da simulação e a saturação de cada via.

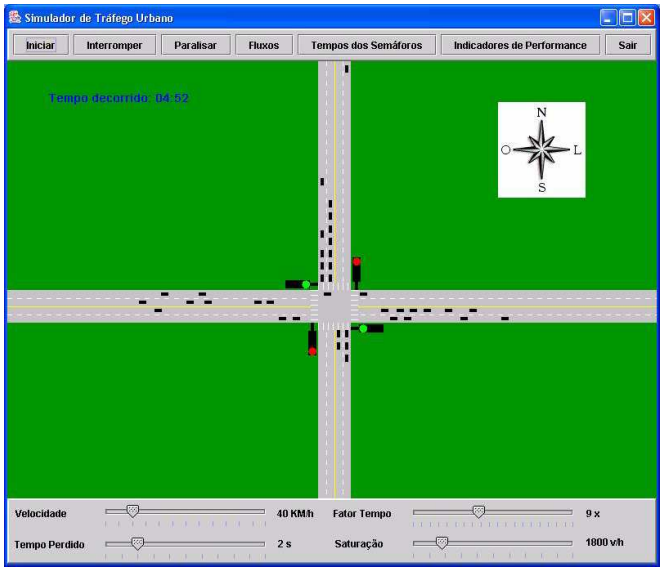

Figura 10: Interface com o usuário

A Figura 11 mostra a *interface* de ajuste dos tempos dos semáforos. Esse ajuste é feito para a via na vertical (Norte/Sul, Sul/Norte) e na horizontal (Leste/Oeste, Oeste/Leste).

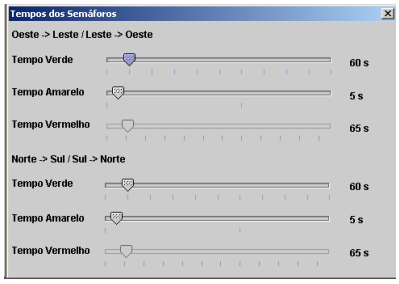

Figura 11: Interface de ajuste dos tempos dos semáforos

A Figura 12 mostra a *interface* de visualização dos indicadores de performance. Os valores dessa figura foram obtidos a partir das entradas contidas na Tabela 2.

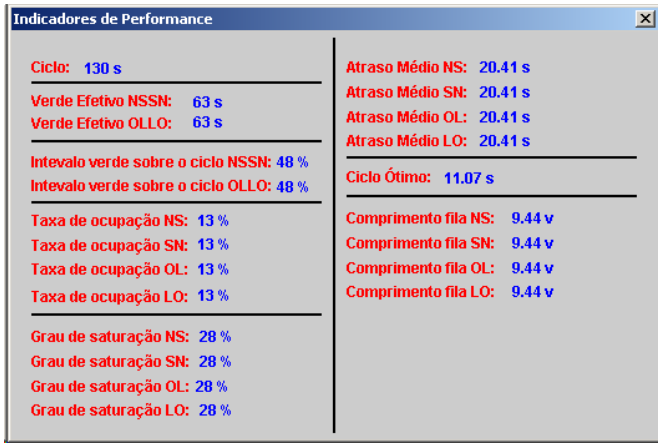

Figura 12: Interface de visualização dos indicadores de performance

$Velocidade(V)$	$40\frac{Km}{h}$
$Tempo\ Perdido(T_p)$	$2s$
$Fator\ Tempo$	$4x(vezes)$
$Saturação(S)$	$16\frac{veículos}{hora}$
$Tempo\ Verde(OLLO))$	$60s$
$Tempo\ Verde(NSSN)$	$60s$
$Tempo\ Amarelo(OLLO)$	$5s$
$Tempo\ Amarelo(NSSN)$	$5s$

Tabela 2: Cálculo da velocidade do veículo

8 CONCLUSÕES

Neste trabalho abordamos a construção de um simulador de tráfego urbano, representando uma interseção semaforizada. O simulador aqui desenvolvido poderá ser utilizado para fins didáticos, uma vez que é apresentada uma gama de elementos pertencentes ao controle de tráfego urbano. Apesar da representação simplificada de uma interseção semaforizada, dados coletados em um ambiente real poderão substituir dados imprecisos inseridos no projeto deste simulador.

Esse simulador não conta com a sofisticação e complexidade dos simuladores utilizados por especialistas da área de engenharia de tráfego. O que não impedirá que o mesmo venha ser aperfeiçoado através de projetos futuros, pois, é exatamente focando este ponto que utilizamos um processo de software. Neste trabalho procuramos descrever os elementos do processo RUP que foram necessários para o desenvolvimento do software. Ao aplicarmos um processo no desenvolvimento acreditamos ter produzido documentação suficiente para que o software possa sofrer aperfeiçoamentos futuros. Em (Akçelik and Besley, 2001) há uma boa explicação sobre a elaboração de simuladores e análise de performance para o estudo de interseções.

Alguns dados utilizados no simulador de tráfego foram inseridos aleatoriamente, sem que houvesse uma busca por modelos ou convenções mais realísticas. Um trabalho pode ser desenvolvido juntamente com profissionais da área de Engenharia de Tráfego, com o intuito de transformar o simulador aqui concebido em uma ferramenta profissional. Uma alternativa é fazer uma coleta de dados em campo e tentar adequar as convenções realizadas neste trabalho para um modelo mais realístico, em (Júnior, 2002) há um técnicas para realização dessas coletas.

Tudo que foi desenvolvido neste trabalho servirá como base para a evolução de um simulador mais sofisticado. Este trabalho poderá ser abordado de duas formas em trabalhos futuros: como um protótipo ou como um primeiro incremento do sistema. À parte de animação gráfica implementada poderá ser utilizada como ponto de apoio para novas implementações, e as expressões implementadas como base para um trabalho interdisciplinar, orientando assim sobre quais dados devem ser obtidos para essa sofisticação. No momento em que a complexidade deste software for aumentando, novos elementos do RUP poderão ser acrescentados no processo de desenvolvimento, isso tende a aumentar o domínio sobre esta metodologia.

Neste trabalho abordamos os elementos do RUP, e expomos o fluxo de trabalho envolvido nas suas diversas fases. Ao longo do processo várias atividades foram desenvolvidas, ao final de cada uma dessas atividades artefatos foram produzidos. Esses artefatos compõem todo o conjunto do software. Neste projeto foram implementados os fluxos de requisito, análise e projeto e implementação, juntamente com as fases de concepção, elaboração e construção. Os outros fluxos de trabalho e a fase de transição podem ser incorporados em possíveis trabalhos futuros.

9 *

Referências

Akçelik, R. and Besley, M. (2001). Microsimulation and analytical methods for modelling urban traffic, *Conference on Advance Modeling Techniques and Quality of Service in Highway Capacity Analysis* .

Banks, J. (1998). *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice*, EMP, New York, USA.

Booch, G., Rumbaugh, J. and Jacobson, I. (2000). *UML, Guia do Usuário*, Editora Campus, Rio de Janeiro, Brasil.

Deitel, P. J. and Deitel, H. M. (2003). *Java Como Programar*, 4 edn, Bookman, Porto Alegre, Brasil.

DENATRAN (1979). *Serviços de Engenharia: Manual de Semáforos*, Ministério da Justiça, Brasília, Brasil.

Iravani, H. and Aashtiani, H. Z. (1999). Use of intersection delay functions to improve realiability of traffic assingment model, *14th Annual International EMME/2 Conference* .

Jain, R. (1991). *The Art of Computer Systems Performance Analysis*, John Wiley & Sons, New York, USA.

Júnior, J. F. N. (2002). *Controle de tráfego urbano: Projeto com apoio de algoritmo genérico e simulação*, Master's thesis, Escola de Engenharia Elétrica e de Computação - UFG, Goiânia, Brasil. 158p.

Kruchten, P. (2000). *The Rational Unified Process: An Introduction*, 2 edn, Addison Wesley.

Lees, J. (2003). A simulation of traffic flow on highways. 57p. Disponível em http://www.stetson.edu/departments/mathcs/students/... ...research/math/ms498/2002/jessica/ final.pdf, último acesso em Junho de 2004.

Leite, G. F. M. (2003). Desenvolvimento de um software científico usando o processo rup: Uma aplicação à simulação de tráfego urbano. Monografia de Projeto Final de Curso, defendida sob a orientação do professor Dr. Antônio César Baleeiro Alves, na UCG, em dezembro de 2003.

Pressman, R. S. (2002). *Engenharia de Software*, 5 edn, McGraw Hill.

Santander, V. F. A. and de Vasconcelos, A. M. (2003). Mapeando o processo unificado em relação ao cmm - nível 2. Disponível em http://www.qualiti.com.br/artigos/RUPxCMM.pdf, último acesso em 30 maio de 2003.

Sommerville, I. (2003). *Engenharia de Software*, 6 edn, Addison Wesley, São Paulo, Brasil.

Treiber, M. and Helbing, D. (1999). Explanation of observed features of self-organization in traffic flow. Disponível em http://xxx.uni-augsburg.de/abs/cond-mat/9901239, último acesso em junho de 2004.