TEN | Traffic Engine

Projeto de Software

**Versão 1.00**

17/06/2009

# Responsáveis

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Antônio Cláudio Goméz de Sousa

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Fernando Seabra Chirigati

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Rafael Shinji Aoki Kikuchi

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Talita Lopes Gomes

# Relatório de Mudanças

TEN – Traffic Engine

**Versão 1.00 – 17/06/2009**

Criação do Documento

# Índice

[1. Introdução 1](#_Toc232501909)

[1.1 Finalidade 1](#_Toc232501910)

[1.2 Escopo 1](#_Toc232501911)

[1.3 Definições, Acronismos e Abreviaturas 1](#_Toc232501912)

[2. Referências 2](#_Toc232501913)

[3. Decomposição 3](#_Toc232501914)

[3.1 Decomposição em Módulos 3](#_Toc232501915)

[3.1.1 Descrição do módulo 1 **Erro! Indicador não definido.**](#_Toc232501916)

[3.2 Decomposição em Processos Concorrentes 3](#_Toc232501917)

[3.2.1 Descrição do processo 1 **Erro! Indicador não definido.**](#_Toc232501918)

[3.3 Decomposição de Dados 4](#_Toc232501919)

[3.3.1 Descrição da entidade de dados 1 4](#_Toc232501920)

[4. Descrição das Dependências 5](#_Toc232501921)

[4.1 Dependência entre módulos 5](#_Toc232501922)

[4.2 Dependência entre processos 5](#_Toc232501923)

[4.3 Dependência entre dados 5](#_Toc232501924)

[4.4 Diagramas de Seqüência 5](#_Toc232501925)

[4.5 Diagrama de Pacotes 6](#_Toc232501926)

[5. Descrição das Interfaces 7](#_Toc232501927)

[5.1 Interfaces dos Módulos 7](#_Toc232501928)

[5.1.1 Interfaces do módulo 1 7](#_Toc232501929)

[5.2 Interfaces entre Processos 7](#_Toc232501930)

[5.2.1 Interfaces do processo 1 7](#_Toc232501931)

[6. Projeto Detalhado 7](#_Toc232501932)

[6.1 Projeto Detalhado dos Módulos 7](#_Toc232501933)

[6.1.1 Detalhamento do módulo 1 7](#_Toc232501934)

[6.2 Projeto Detalhado das Entidades de Dados 7](#_Toc232501935)

[6.2.1 Detalhamento da entidade de dados 1 7](#_Toc232501936)

# Introdução

## Finalidade

Este documento visa especificar os módulos, os processos e as estruturas de dados pertencentes ao TEN – Traffic Engine –, cujo ciclo de vida segue o Modelo de Processo em Cascata. Dessa maneira, forma-se uma base mais detalhada para o desenvolvimento do software, ajudando a equipe na atividade de codificação.

Além da equipe de desenvolvimento do sistema, a audiência deste documento é composta pelo gerente de qualidade do projeto, o professor Antônio Cláudio Goméz de Sousa.

## Escopo

O TEN – Traffic Engine – tem como principal objetivo a simulação de tráfego urbano em uma região confeccionada pelo próprio usuário através da interface. Logo, o software deve prover todas as ferramentas necessárias ao desenho dessa região, assim como todas as funcionalidades relacionadas à simulação. Além disso, a geração de um relatório referente a essa simulação pertence ao escopo do sistema.

## Definições, Acronismos e Abreviaturas

Não se aplica.

# Referências

* *Plano de Gerenciamento de Projeto de Software – PGPS*

Versão: 1.00

Data: 01/04/2009

Responsável: Equipe de desenvolvimento do projeto

* *Especificação de Requisitos de Software – ERS*

Versão: 1.00

Data: 18/05/2009

Responsável: Equipe de desenvolvimento do projeto

* *Manual do Usuário – 2ª versão*

Versão: 1.00

Data: 17/06/2009

Responsável: Equipe de desenvolvimento do projeto

# Decomposição

## Decomposição em Módulos

O aplicativo será dividido em cinco módulos: simulador, interface, atualizador, dados do mapa e dados dos veículos. Abaixo se encontra um diagrama que representa essa divisão.



A seguir, cada um dos módulos será descrito.

### Simulador

Kiks ama a namorada dele. A namorada dele o ama. Logo, eles são felizes =D

### Interface

asds

### Atualizador

asdsd

### Dados do Mapa

asdsd

### Dados dos Veículos

asdasd

## Decomposição em Processos Concorrentes

O sistema pode ser dividido em três processos concorrentes: o primeiro é responsável pela interface, o segundo pela simulação e o último deles encarrega-se do controle de redesenho da interface durante a execução de uma simulação. Cada um desses processos é melhor explicado abaixo.

### Interface

Este processo é responsável pela manutenção da interface gráfica e por cada evento disparado de suas possíveis interações com o usuário.

### Simulação

Este processo é responsável pela aplicação de algoritmos de simulação sobre os dados do mapa e dados dos veículos.

### Controle de redesenho da interface

Este processo é responsável pelo controle de redesenho da interface gráfica durante uma simulação: nem toda iteração da simulação implica em um redesenho. Este controle é realizado com intuito de reduzir o custo computacional do sistema evitando uma sobrecarga de redesenhos.

## Decomposição de Dados

Em OO, a decomposição de dados já é feita automaticamente com a decomposição em módulos, através da UML, por exemplo. Logo, basta, nesse item, referenciar o item 3.1. Além disso, referenciar também o **Dicionário de Dados** do ERS.

# Descrição das Dependências

## Dependência entre módulos

Nano, deixar os itens abaixo nos trinques:

* Interface depende dos módulos de dados para reproduzir a simulação.
* Atualizador depende do simulador pra saber se ele deve ou não enviar redesenhos para a interface (simulação rodando ou não).
* Simulador depende dos módulos de dados para aplicar os algoritmos de simulação.

## Dependência entre processos

Não há. Só tem aquela dependência do atualizador com o simulador que disse no item acima, mas, fora isso, são independentes. Talvez seja importante dizer que há compartilhamento de dados em comum (tanto dados do mapa quanto dados dos veículos) e que, por isso, é necessário um controle de concorrência – só entre simulador e interface.

## Dependência entre dados

A dependência entre os dados é mostrada anteriormente na decomposição do sistema em módulos, que se encontra no item 3.1 deste documento. O diagrama de classes presente no item 3.2.5 da Especificação de Requisitos de Software também apresenta informações relevantes à dependência entre os dados.

[Descrever os relacionamentos entre os dados. Mais uma vez, em OO, a dependência entre dados é descrita juntamente com os módulos, não sendo necessário nesse item, portanto, repetir a análise dessa dependência; basta referenciar. Referenciar também o **Diagrama de Classes** do ERS.]

## Diagramas de Seqüência

O diagrama de seqüência mostra claramente como as classes colaboram de maneira a executar um determinado caso de uso. Na Especificação de Requisitos de Software, diagramas de seqüência referentes a três estudos de casos, os quais também estão especificados no mesmo documento, foram apresentados, porém sem considerar as classes e os métodos do sistema.

Nesta seção, os mesmos diagramas são mostrados, mas agora contendo maiores detalhes em relação às classes e aos métodos.

#### Caso de Uso 1 – Criar Rua

#### Caso de Uso 7 – Semaforizar um Ponto



#### Caso de Uso 11 – Iniciar Simulação



## Diagrama de Pacotes

O diagrama de pacotes é importante para descrever como os sub-sistemas do software são divididos em agrupamentos lógicos de classes e objetos coesos, mostrando as dependências existentes.

O diagrama de pacotes do TEN encontra-se abaixo.



# Descrição das Interfaces

## Interfaces dos Módulos

### Interfaces do módulo 1

Interfaces de usuários, externas e internas.

Não tem interfaces externas. Interfaces internas = métodos públicos.

## Interfaces entre Processos

A comunicação entre os processos é feita através dos módulos expostos acima.

### Interfaces do processo 1

Interfaces do processo 1 com os demais processos.

# Projeto Detalhado

## Projeto Detalhado dos Módulos

Nano, se puder adiantar os métodos que estão nos diagramas de sequência (e fazer os similares)...

### Detalhamento do módulo 1

Descrição da lógica e das estruturas de dados do módulo 1.

## Projeto Detalhado das Entidades de Dados

### Detalhamento da entidade de dados 1

Modelo físico da entidade de dados 1.