Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Colegiado de Sistemas de Informação

DengueME: Um framework de software para modelagem da Dengue e seu vetor

Lucas Saraiva Ferreira

TRABALHO DE CONCLUSÃO DE CURSO

ORIENTAÇÃO:
Tiago França Melo de Lima

Março, 2017
João Monlevade/MG

Lucas Saraiva Ferreira

DengueME: Um framework de software para modelagem da Dengue e seu vetor

Orientador: Tiago França Melo de Lima

Monografia apresentada ao curso de Sistemas de Informação do Departamento de Computação e Sistemas da Universidade Federal de Ouro Preto como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação

Universidade Federal de Ouro Preto

João Monlevade

Março de 2017

UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E APLICADAS
COLEGIADO DO CURSO DE SISTEMAS DE INFORMAÇÃO

FOLHA DE APROVAÇÃO DA BANCA EXAMINADORA

DengueME: Um framework de software para a modelagem da Dengue e seu vetor.

Lucas Saraiva Ferreira

Monografia apresentada ao Instituto de Ciências Exatas e Aplicadas da Universidade Federal de Ouro Preto como requisito parcial da disciplina CSI499 – Trabalho de Conclusão de Curso II do curso de Bacharelado em Sistemas de Informação e aprovada pela Banca Examinadora abaixo assinada:

Prof. Me. Tiago França de Melo de Lima
DECSI - UFOP

Prof. Dr Leonardo Vieira dos Santos Reis
DECSI - UFOP

Prof. Me. Thiago Luange Gomes
DECSI - UFOP

João Monlevade, 23 de março de 2017

Rua Trinta e seis, 115 – Bairro Loanda – CEP 35931-008 – João Monlevade – MG – Brasil
http://www.icea.ufop.br – cosi@decea.ufop.br – (31) 3852-8709

UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E APLICADAS
COLEGIADO DO CURSO DE SISTEMAS DE INFORMAÇÃO

ATA DE DEFESA

Aos 23 dias do mês de março de 2017, às 17 horas e 20 minutos, na sala H 102 do Instituto de Ciências Exatas e Aplicadas, foi realizada a defesa de Monografia pelo aluno **Lucas Saraiva Ferreira,** sendo a Comissão Examinadora constituída pelos professores: Prof. Me. Tiago França de Melo Lima, Prof. Me. Thiago Luange Gomes e Prof. Dr. Leonardo Vieira dos Santos Reis.

O candidato apresentou a monografia intitulada: *"DengueME: Um framework de software para a modelagem da Dengue e seu vetor".* A comissão examinadora deliberou, por unanimidade, pela aprovação do candidato, com nota 10 (DEZ PONTOS), concedendo-lhe o prazo de 15 dias para incorporação das alterações sugeridas ao texto final.

Na forma regulamentar, foi lavrada a presente ata que é assinada pelos membros da Comissão Examinadora e pelo graduando.

João Monlevade, 23 de março de 2017.

Prof. Me. Tiago França de Melo de Lima
Professor Orientador/Presidente

Prof. Dr.Leonardo Vieira dos Santos Reis
Professor Convidado

Prof. Me. Thiago Luange Gomes
Professor Convidado

Lucas Saraiva Ferreira
Graduando

UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E APLICADAS
COLEGIADO DO CURSO DE SISTEMAS DE INFORMAÇÃO

TERMO DE RESPONSABILIDADE

Eu, Lucas Ignacio Ferreira, declaro que o texto do trabalho de conclusão de curso intitulado "DengueME: Um framework de software para o modelagem da Dengue e seu vetor." é de minha inteira responsabilidade e que não há utilização de texto, material fotográfico, código fonte de programa ou qualquer outro material pertencente a terceiros sem as devidas referências ou consentimento dos respectivos autores.

João Monlevade, 5 de Abril de 2017

Lucas Ignacio Ferreira
Assinatura do aluno

Este trabalho é dedicado a todos que desejam fazer a diferença, mesmo que nas pequenas coisas.

Agradecimentos

Agradeço a todos que influenciaram ou participaram diretamente ou indiretamente na criação deste trabalho.

"You did everything to bury me,
but you forgot that I was a seed."
(Dinos Christianopoulos)

Resumo

A dengue é um grande desafio para a saúde pública e possui uma complexa dinâmica de transmissão. Diversos fatores influenciam essa dinâmica, tais como falta de infraestrutura urbana, densidade populacional elevada, mobilidade (vetorial e humana) e condições microclimáticas. A modelagem matemática e simulação computacional são ferramentas úteis para o estudo de problemas dessa natureza. Elas têm sido empregadas em diversas pesquisas relacionadas ao tema, como por exemplo, para analisar o efeito da temperatura sobre a ecologia do vetor, simular a disseminação do vírus em um ambiente urbano e avaliar custo-efetividade de diferentes estratégias de vacinação. O DengueME (*Dengue Modeling Environment*) é uma plataforma para apoiar o estudo da dinâmica da dengue por meio da modelagem e simulação computacional. Para facilitar o uso dos modelos por pessoas sem muita experiência em programação, ele oferece uma interface gráfica amigável que auxilia na parametrização dos modelos e o código fonte correspondente é gerado automaticamente. E auxilia usuários mais avançados, modeladores, a especificar a interface gráfica de parametrização dos seus modelos, facilitando sua disponibilização para outras pessoas. No entanto, a versão preliminar inicialmente disponibilizada apresentava diversos problemas, tais como: baixa usabilidade, grande limitação de funcionalidades para a especificação de interfaces gráficas, suporte a apenas uma linguagem de modelagem/programação. O trabalho tem como objetivo identificar e desenvolver as correções e melhorias necessárias para disponibilizar publicamente a versão 1.0 do software DengueME. Para isso, os seguintes passos foram realizados: 1) refatoração do código-fonte, 2) avaliação "rápida e suja"da interface gráfica, 3) ciclos curtos de concepção, (re)design, prototipação/codificação, testes, avaliação. Os resultados obtidos incluem refatoração do código-fonte, identificação e correção de problemas de usabilidade, atualização do suporte e integração com a plataforma de modelagem TerraME, atualização dos modelos disponíveis para torná-los compatíveis com a versão mais recente do TerraME e com a interface gráfica do editor de modelos, disponibilização da biblioteca de modelos em um servidor, documentação de usuário e desenvolvedor, publicação de versão. Os trabalhos futuros incluem desenvolver novos modelos e oferecer suporte para a linguagem de programação R.

Palavras-chaves: dengue, *Aedes aegypti*, modelagem, simulação, DengueME, TerraME.

Abstract

Dengue is a great challenge to public health and it has a complex transmission dynamic. Several factors can affect this dynamic, such as the lack of urban infrastructure, high population density, mobility and microclimate conditions. Mathematical modelling and computer simulation is a tool that can be used to study problems of this nature. Modelling has been applied in many related researchs such as to analyze the effects of the temperature on vector development, to simulate the spread of the virus in an urban environment, and to evaluate different vaccination strategies. DengueME (Dengue Modeling Environment) is a platform that aims to support the study of the dengue dynamics using modeling and computer simulation. DengueME aims to help people without great experience in programming to use models. It offers a friendly graphical user interface allowing users to parametrize models and run simulations. Advanced users can use DengueME to create graphical interfaces to theirs models and share this interface with other users. However, the beta version of the tool had some bugs and interface issues such as: low usability level, limited features for interface specification, and support to only one modeling and simulation platform. In this context, this work aimed to identify and correct problems, develop new features, and release a new version. To accomplish this, the following steps were performed: refactoring of the source-code; informal evaluation of the graphical user interface; short cycles of (re) design, prototyping/coding, testing and evaluation. The obtained results include the source code refactored, improvements in usability, update of TerraME version supported, availability of the models library in a hosted version, user and developer documentation. Future works include the development of new models and the support to R programming language.

Key-words: dengue. software. modeling. DengueME.

Lista de ilustrações

Lista de abreviaturas e siglas

API	Application Programming Interface
DengueME	Dengue Modeling Environment
IDE	Integrated Development Environment
IHC	Interação Humano-Computador
TerraME	Terra Modeling Environment
STEM	Spatiotemporal Epidemiological Modeler
XML	eXtensible Markup Language
WHO	World Health Organization

Sumário

1 Introdução

A dengue é uma arbovirose transmitida por mosquitos da família *Aedes*. O mosquito *Aedes aegypti* é o principal vetor da doença no Brasil, onde ela tem sido endêmica por mais de 30 anos (CâMARA et al., 2009) e é um grande desafio para a saúde pública. A primeira epidemia de grandes proporções registrada pelo ministério da saúde aconteceu entre 1986 e 1987, na cidade do Rio de Janeiro (NOGUEIRA et al., 1999). Em 2015, foram registrados cerca de 1.650.000 casos prováveis e aproximadamente 860 óbitos, o que corresponde a um aumento de 82,5% nos casos de morte em relação ao ano de 2014 (SVS-BRASIL, 2016). Além da dengue, o mosquito *Aedes Aegypti* é também transmissor dos vírus Chikungunya, Zika e febre amarela urbana. Compreender a dinâmica espaço-temporal de transmissão da doença e/ou ecologia do vetor é um problema complexo, que carece de ferramentas para apoiar seu estudo.

A modelagem e simulação matemática-computacional é uma ferramenta interessante para estudar fenômenos complexos e pode apoiar atividades de planejamento, avaliação de cenários e tomada de decisão. De acordo com Brown (2009), a grande facilidade de locomoção nos tempos atuais e mudanças no meio ambiente tem levantado uma demanda em estudos sobre difusão de doenças. O aumento do poder computacional associado à redução dos seus custos contribuiu para a ampliação do uso da modelagem e simulação no estudo de sistemas dinâmicos (SOKOLOWSKI; BANKS, 2010). Por exemplo, identificar padrões espaço-temporais de transmissão de doenças como a dengue, que apresenta uma dinâmica complexa, tanto em relação à transmissão do vírus quanto à ecologia de seu vetor, que é influenciada por um amplo conjunto de fatores (ex. meteorológicos, ambientais, sociais).

Ferramentas que auxiliam na modelagem e simulação de sistemas complexos são bastante úteis e podem ter como intuito apoiar desde usuários básicos até avançados. Neste contexto, o DengueME (*Dengue Modeling Environment*) é uma plataforma voltada para o estudo da dinâmica da Dengue e de seu vetor por meio da modelagem e simulação computacional (LIMA et al., 2016). A ferramenta oferece uma interface amigável e de fácil entendimento para usuários não experientes em programação. Através da sua interface, o usuário pode parametrizar os modelos, gerar o código-fonte correspondente e executar simulações. Para usuários mais avançados, por exemplo modeladores, a ferramenta oferece um módulo especial denominado *Model Builder*, que auxilia na construção de interfaces gráficas para parametrização de modelos, facilitando a disponibilização e utilização deles.

No entanto, a versão preliminar da ferramenta apresentava diversos problemas, tais como: baixa usabilidade, limitação de funcionalidades para a especificação de interfaces

gráficas, uso de versão desatualizada do *framework* QT (QTCOMPANY, 2017), suporte a versão desatualizada da plataforma de modelagem TerraME (CARNEIRO et al., 2013; TERRALAB, 2017), suporte a uma única linguagem de modelagem/programação, e indisponibilidade de recursos para desenvolvimento de uma biblioteca de modelos.

1.1 Objetivos

Neste contexto, este trabalho tem como objetivo geral implementar as correções e melhorias necessárias na versão mais recente disponível do DengueME para produzir uma versão estável e disponibilizá-la publicamente. O escopo do projeto inclui refatoração, concepção, projeto, implementação, testes e documentação de software. Fazem parte dos objetivos específicos do trabalho:

1. refatorar o código-fonte;
2. implementar melhorias na qualidade de uso (usabilidade) da interface gráfica;
3. atualizar os modelos atualmente disponíveis na plataforma TerraME versão 1.6 para a versão 2.0.4;
4. implementar um editor de modelos, que permita ao usuário adicionar novos modelos (em TerraML) à biblioteca do DengueME e configurar a interface gráfica para parametrização dos modelos;
5. disponibilizar a biblioteca de modelos em um servidor remoto, permitindo que ela seja atualizada de forma independente das atualizações do software;
6. elaborar documentação de desenvolvedor e de usuário;
7. empacotar e distribuir o software DengueME versão 1.0, incluindo documentação e código fonte, para ambientes Linux e Windows.

1.2 Organização do Trabalho

O texto está organizado em quatro capítulos adicionais, além do capítulo Introdução. No segundo capítulo, Conceitos Básicos e Trabalhos Relacionados, são apresentadas informações necessárias ao entendimento do trabalho, bem como ferramentas e técnicas utilizadas para sua realização. O terceiro capítulo, Metodologia, descreve as etapas realizadas no desenvolvimento desse trabalho. No quarto capítulo, Resultados, são apresentados os resultados obtidos em cada uma das etapas realizadas, e apresenta versão do DengueME que foi desenvolvida. Por fim, o último capítulo apresenta as considerações finais e os trabalhos futuros.

2 Conceitos Básicos e Trabalhos Relacionados

Neste capítulo são apresentados alguns conceitos básicos necessários ao entendimento do texto, bem como alguns trabalhos correlatos.

2.1 Modelagem e Simulação da Dengue

Um sistema é constituído por um conjunto de entidades, dependentes ou independentes, reais ou abstratas, inseridas em um ambiente e relacionadas entre si de alguma forma (BAIANU, 2011). Em geral, é difícil compreender sistemas complexos apenas pelo estudo de suas entidades internas, pois a interação entre elas leva à propriedades e comportamentos emergentes (BAIANU, 2011). A modelagem e simulação computacional é interessante para lidar com tais problemas.

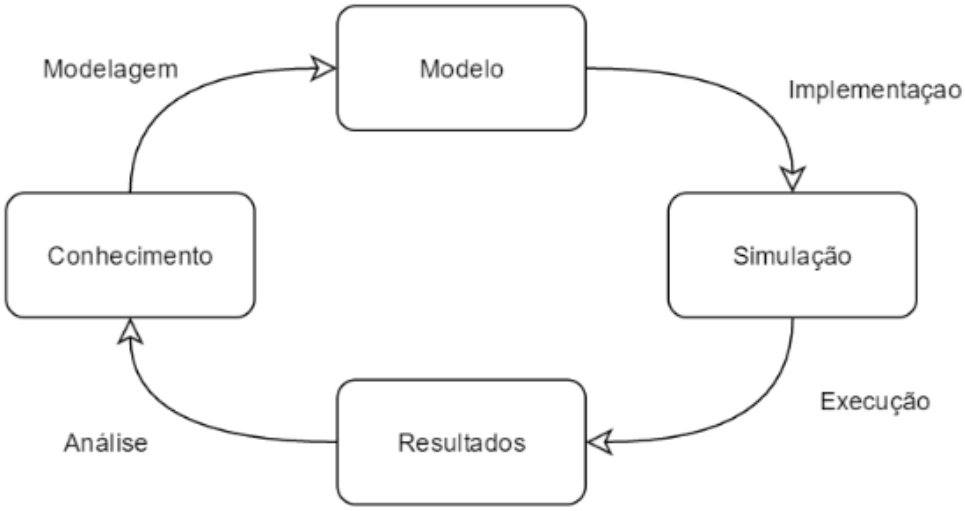
A modelagem é uma técnica amplamente utilizada para ajudar a compreender sistemas complexos. Um modelo pode ser visto como uma representação simplificada da realidade (sistema em estudo) (SOKOLOWSKI; BANKS, 2010). Para se criar um modelo é necessário abstrair detalhes que não sejam relevantes, e se concentrar nas propriedades e/ou interações que sejam mais relevantes para a compreensão do objeto de estudo (SOKOLOWSKI; BANKS, 2010). É impraticável, ou mesmo impossível Não é possível incluir em um modelo todas variáveis e interações existentes em um sistema complexo.

Modelos podem ter diversos usos além da simulação para predição de cenários. Epstein (2008) apresenta as motivações para se usar a modelagem, e ilustra uma série de outros possíveis usos que modelos podem ter. Um exemplo, diferente da predição, seria utilizar a modelagem para auxiliar no planejamento e tomada de decisões em saúde pública. Outra forma de uso seria utilizar modelos para explicar comportamentos ou descobrir novos questionamentos sobre os comportamentos ali representados.

Após especificar o modelo, o que pode ser feito em uma linguagem de programação, é possível simular o sistema ou fenômeno por ele representado. A simulação consiste na execução do modelo, que gera uma imitação dos acontecimentos reais do sistema dentro de um determinado intervalo temporal (SOKOLOWSKI; BANKS, 2010). Através da simulação, é possível criar cenários hipotéticos, por exemplo, fornecendo diferentes parâmetros de entrada, e analisando o comportamento e os resultados que serão gerados como saída da execução do modelo. Assim, um pesquisador/modelador pode analisar os resultados provenientes de seu modelo, e melhorá-lo até que atinja o objetivo desejado.

Várias atividades podem ser empregadas em processos de modelagem e simulação para estudar fenômenos complexos. Mas de forma geral, pode-se dizer que é um processo cíclico, que envolve atividades de concepção (implementação do modelo), simulação (execução do modelo), resultados (geração e análise), e construção de conhecimento (SOKOLOWSKI; BANKS, 2010), conforme ilustrado na Figura 1.

Figura 1 – Clico de Modelagem



Fonte: (SOKOLOWSKI; BANKS, 2010)

2.2 Plataformas de Modelagem e Simulação

É possível encontrar na literatura uma grande quantidade de ferramentas para modelagem e simulação de sistemas e/ou fenômenos. Elas se diferenciam em diversos aspectos, como por exemplo pelos tipos de paradigmas/modelos suportados ou mesmo o foco da ferramenta em si. Algumas delas são apresentadas nessa seção. Não foi possível encontrar nenhuma ferramenta com foco em modelagem e simulação da Dengue em escala intra-urbana.

O NetLogo (TISUE; WILENSKY, 2004) é um ambiente gráfico para modelagem e simulação de fenômenos naturais e sociais. Através de sua interface gráfica, é possível criar, parametrizar e simular diferentes modelos. Ele usa a linguagem Logo para especificar as regras de comportamento do modelo. Um grande diferencial do NetLogo é sua extensa biblioteca de modelos, que possui grande variedade de exemplos em diferentes áreas de conhecimento (ex. epidemiologia, ciências sociais, ecologia, urbanismo). Por outro lado, a ferramenta exige do usuário conhecimentos na linguagem Logo para o desenvolvimento de modelos. Cada modelo possui sua própria interface gráfica de parametrização e simulação, que é construída utilizando-se a interface da ferramenta. O usuário tem liberdade para

organizar os elementos de interface do seu modelo da forma que julgar melhor - flexibilidade que tanto pode ser um aspecto positivo quanto negativo.

O TerraME (*Terra Modelling Environment*) (TERRALAB, 2016) é uma plataforma *open source* para modelagem e simulação espaço-temporal das interações natureza-sociedade. Ele oferece suporte a diferentes paradigmas de modelagem: autômatos celulares, agentes e redes. Os modelos são implementados na linguagem TerraML (*Terra Modelling Language*), uma linguagem de domínio especifico derivada da linguagem Lua, através da qual é possível descrever o comportamento e as regras do sistema. A plataforma oferece suporte para a modelagem de diferentes tipos de fenômenos, como sistemas ambientais, interações sociais, e espalhamento de doenças. A linguagem TerraML oferece diversas estruturas de dados e serviços, como por exemplo, para visualização dos resultados da simulação - Chart (gráfico), LogFile (arquivo csv), VisualTable (tabela) e TextScreen (tabela).

O Epifire é uma API (*Application Programming Interface*) e interface gráfica de usuário para criação, análise e manipulação de redes voltada para modelagem e simulação epidemiológica. Ela oferece um *framework* flexível para criação e análise de redes epidemiológicas de contato e permite simular a transmissão de doenças através dessas redes (HLADISH et al., 2012). Apesar de oferecer uma interface gráfica, para utilizá-la adequadamente é necessário algum conhecimento em programação para especificar algumas partes da rede de contato (HLADISH et al., 2012). Através da interface gráfica é possível parametrizar a rede e a simulação, e visualizar os resultados por meio de em gráficos e logs. Um recurso interessante oferecido pela ferramenta é o módulo que auxilia na análise dos resultados, realizando automaticamente cálculos estatísticos relacionados aos parâmetros de entrada e de saída (HLADISH et al., 2012).

O MalariaTool (GRIFFIN et al., 2010) é uma interface gráfica para a parametrização e simulação de um modelo de intervenção específico para a Malária, causada pelo protozoário *Plasmodium falciparum*. Detalhes sobre o modelo podem ser encontrados em Griffin (2010). A versão mais recente da ferramenta, v. 3.2, foi disponibilizada em 2015 (GRIFFIN; STOYANOV, 2015). Uma limitação é a possibilidade de se usar um único modelo e o fato que a interface gráfica precisa sempre refletir as atualizações no modelo, além de alguns problemas de usabilidade. A ferramenta oferece recursos para geração e visualização dos resultados, como por exemplo gráficos, que podem ser customizados.

O STEM (*Spatiotemporal Epidemiological Modeler*) (ECLIPSE, 2017) é uma plataforma *standalone* (RCP, *Rich Client Platform*) baseada no Eclipse (http://www.eclipse.org). A ferramenta tem o intuito de apoiar modeladores na criação e uso de modelos espaciais e temporais de doenças infecciosas. O STEM é uma ferramenta madura e completa, que possui uma variedade de modelos prontos. Um desafio consiste em sua utilização por pessoas não familiarizadas com a complexa interface gráfica do Eclipse e o uso de ambientes de desenvolvimento integrado (IDE, do inglês *Integrated Development Environment*).

2.3 DengueME

O DengueME (*Dengue Modeling Environment*) é um *framework* de software para modelagem e simulação da Dengue e da ecologia do seu vetor (LIMA et al., 2014; LIMA et al., 2016). Modelos computacionais podem ser utilizados para melhor compreender a dinâmica de transmissão da dengue e da ecologia do seu vetor, e cenários podem ser criados para analisar o impacto de intervenções. A quantidade de artigos relacionados a modelos sobre dengue vem crescendo ao longo dos últimos anos, o que sugere uma oportunidade para o desenvolvimento de ferramentas que apoiem a criação e o uso desses modelos.

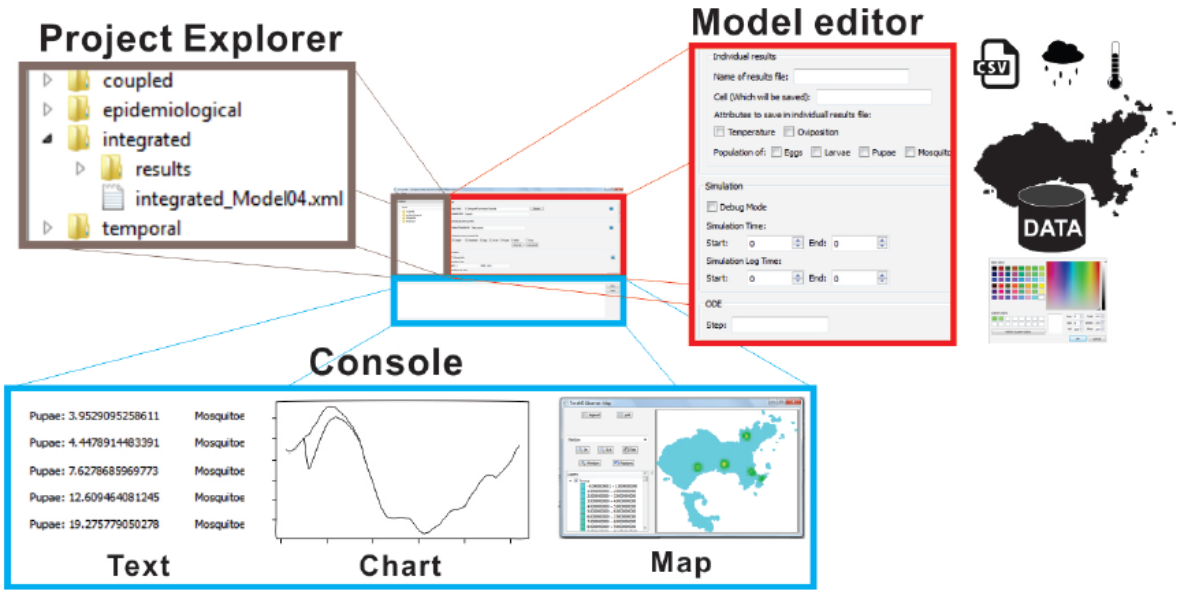
O DengueME oferece uma interface gráfica simples e amigável para auxiliar o usuário na criação, parametrização, simulação de modelos, e análise dos resultados. Ele permite ao usuário utilizar modelos disponíveis em sua biblioteca de modelos e também criar novos modelos, incluindo interfaces próprias para parametrização. Os parâmetros do modelo podem ser modificados através da sua interface gráfica, e o código-fonte correspondente é gerado/atualizado automaticamente. A simulação e visualização inicial dos resultados também pode ser feita diretamente pela interface gráfica da ferramenta. Antes do início desse trabalho, a ferramenta implementada usava o Qt versão 4.8 e os modelos o TerraME versão 1.6, ambos bastante desatualizados.

Para usuários avançados, que queiram criar seus próprios modelos e respectivas interfaces gráficas de parametrização, o DengueME oferece um módulo denominado *Model Builder*. Ele permite ao usuário modelador criar suas próprias interfaces gráficas para configuração dos parâmetros do seu modelo. O modelador pode por exemplo definir quais parâmetros poderão ser visualizados/editados pelos usuários finais e o que será apresentado como resultado da simulação e como isso pode ser customizado pelos usuários. Tudo isso é feito pela interface gráfica do *Model Builder* - somente a edição do código-fonte dos modelos precisa ser feita em um editor externo ou ambiente de programação. A configuração da interface gráfica é armazenada em um arquivo XML, que pode ser compartilhado juntamente com os arquivos do modelo, para que possa ser utilizado por outros usuários.

Uma visão geral do ambiente visual de desenvolvimento do DengueME (DengueME VDE - *DengueME Visual Development Environment*) é apresentado na Figura 2. Na *view Project Explorer*, o usuário pode visualizar os projetos e modelos disponíveis no seu diretório de trabalho (*workspace*). Os parâmetros do modelo podem ser configurados na *view Model Editor*, e a *view Console* apresenta a saída gerada pelo interpretador durante a execução do modelo (simulação).

O processo de modelagem utilizando o DengueME VDE é ilustrado na Figura 3. Inicialmente o usuário deve criar um novo projeto (diretório). Depois, ele deve criar um novo cenário (modelo) a partir dos modelos existentes na biblioteca do DengueME. Então,

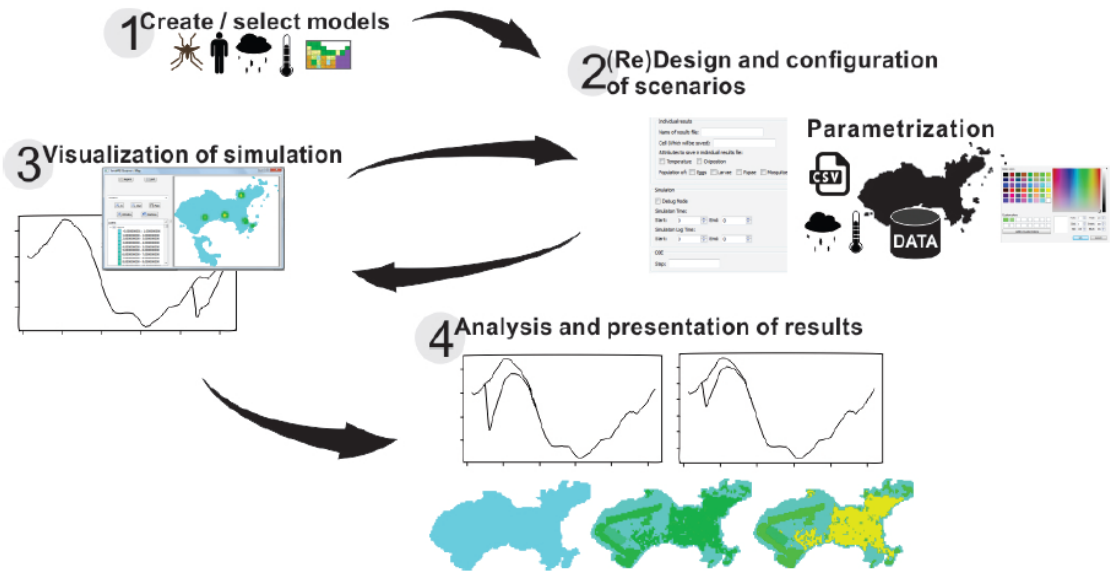
Figura 2 – Visão geral do ambiente visual de desenvolvimento do DengueME (DengueME VDE)



Fonte: (LIMA et al., 2014)

o cenário poderá ser configurado e os parâmetros alterados utilizando o *Model Editor*. Por fim, é possível executar o modelo e visualizar os resultados da simulação. Esse processo pode ocorrer de forma iterativa, conforme necessário, e ao final, os resultados podem ser analisados e reportados.

Figura 3 – Diagrama ilustrando o processo de modelagem utilizando o DengueME VDE



Fonte: (LIMA et al., 2014)

2.4 Design, Desenvolvimento e Avaliação de Software

O design de sistemas interativos, como por exemplo o de uma ferramenta para modelagem e simulação, tem o intuito de projetar e construir sistemas computacionais que sejam agradáveis de usar, que cumpram algum objetivo e que apoiem o usuário em alguma tarefa (BENYON, 2005). Para tanto, se faz necessário levar em conta tanto a tecnologia quanto o usuário como parte fundamental do processo (BENYON, 2005). Desenvolver um sistema que seja funcional mas não proporcione uma boa qualidade / experiência de uso pode ser considerado um fracasso parcial.

Portanto, avaliar a qualidade de um sistema é essencial. Para isso é possível utilizar métodos e técnicas da área de Interação Humano-Computador (IHC). IHC é uma das diversas áreas da ciência da computação. Ela é focada no projeto, implementação e avaliação de sistema computacionais interativos para uso de seres humanos (HEWETT et al., 1992), e se preocupa portanto, com a qualidade de uso e da experiência de uso oferecida por sistemas de computação. Dentre os benefícios de sistemas com boa qualidade de uso, podemos citar fatores como (BARBOSA; SILVA, 2010b): (i) aumento da produtividade do usuário através de uma interface eficiente; (ii) redução de erros cometidos pelo usuário; (iii) aumento da satisfação do usuário ao interagir com o sistema.

Os métodos de avaliação podem ser classificados em três grupos (BARBOSA; SILVA, 2010b):

- Métodos de investigação, envolvem perguntar diretamente aos usuários sobre requisitos, preferências, problemas enfrentados e percepção de qualidade, dentre outros aspectos do sistema que está sendo avaliado. São exemplos a aplicação de questionários, a realização de entrevistas e de grupos focais.

- Métodos de observação, através dos quais é possível coletar dados, de forma direta e/ou indireta, sobre a interação do usuário com o sistema sendo avaliado enquanto ele é utilizado. Tais métodos também podem ser aplicados num estágio inicial, antes de o sistema começar a ser desenvolvido, por exemplo observando como potenciais usuários realizam suas atividades sem o sistema, que ainda será construído. A observação pode usar diversas técnicas para coleta de dados, tais como anotações, gravações em áudio/vídeo, gravação de log da interação.

- Métodos de inspeção, não envolvem a participação dos usuários do sistema no processo de avaliação. Em tais métodos, a avaliação é realizada por especialistas no domínio/tecnologia (ex. dispositivos vestíveis) ou em aspectos que estão sendo avaliados (ex. usabilidade, acessibilidade). Os métodos em geral consistem em inspecionar a interface/interação com o sistema, com o apoio de diretrizes ou *guidelines*, visando a identificação de problemas de design e a indicação de como resolvê-los.

Portanto, é importante incorporar métodos e técnicas de IHC no projeto e desenvolvimento de um software. Após traçar os objetivos iniciais e projeto da ferramenta, dá-se inicio à etapa de prototipação/codificação. O contato com o usuário e revisão dos requisitos do sistema deve ocorrer durante todo o processo de criação e teste da ferramenta. Metodologias de desenvolvimento ágil podem ser empregadas para obtenção rápida de *feedbacks*. Testes devem ser feitos a cada ciclo de desenvolvimento para garantir o atendimento dos requisitos, em termos de funcionalidade e de qualidade de uso.

Nesse trabalho, algumas ferramentas e técnicas foram empregadas com o intuito de melhorar o processo de desenvolvimento do software DengueME. O GitHub (GITHUB, 2017) foi usado para versionamento, controle de mudanças e elaboração da documentação (de usuário e de desenvolvedor). Para criar novos componentes de interfaces e atualizar aqueles já existentes (ex. janelas, *wizards*), protótipos de baixa fidelidade foram desenvolvidos utilizando a ferramenta *Mockups* (MOCKUPS, 2017). O processo de desenvolvimento ocorreu com base no *workflow* apresentado na seção 3.3, que envolve ciclos curtos de (re)design, desenvolvimento e teste/avaliação.

3 Metodologia

Nesse capítulo são apresentadas as etapas realizadas no desenvolvimento do trabalho, que incluem: refatoração do código-fonte, avaliação de usabilidade, desenvolvimento e documentação de software.

3.1 Refatoração

Tendo em vista a existência de uma versão preliminar do software DengueME, a primeira etapa desse trabalho consistiu em realizar uma refatoração do seu código-fonte. A refatoração é um "mal necessário", e de certa forma, uma prática comum no processo de desenvolvimento de software. De acordo com Fowler (1999), a refatoração consiste numa técnica controlada para melhorar o design do código-fonte sem afetar o resultado final (a saída do programa), e ainda que sejam estas melhorias pequenas, em conjunto, elas se tornam significativas.

Ao iniciar esse trabalho, percebeu-se que o código-fonte do DengueME apresentava diversos problemas, tais como: ausência de comentários necessários para a documentação/compreensão do software, presença de comentários e código desnecessário, código obsoleto/não utilizado. Além disso, todas as funcionalidades da ferramenta foram inspecionadas com o intuito de encontrar *bugs* e identificar possíveis melhorias.

Outro ponto importante era a dependência/uso de versões desatualizadas de bibliotecas e APIs. Os modelos disponíveis no DengueME estavam implementados em uma versão desatualizada plataforma TerraME (v. 1.6[1]). Além disso, a versão preliminar do software utilizava o *framework* Qt 4.8, também bastante desatualizada. Dessa forma, optou-se por atualizar o código-fonte dos modelos e do software DengueME para utilizar as versões mais recentes do Qt (Qt 5.7) e TerraME (2.0-beta-4[2]), disponíveis na época do início desse trabalho. Diversas mudanças foram necessárias para que os modelos e o software DengueME continuassem se comportando como esperado, apenas para manter funcional os recursos já existentes. A lista de mudanças entre as versões do *framewok* Qt pode ser encontrada na sua documentação oficial[3].

Além do código-fonte principal do software (C++, Qt), arquivos auxiliares gerados pela ferramenta também foram refatorados, tais como arquivos XML, correspondentes à interface dos modelos e arquivos Lua, correspondentes à configuração de entrada dos modelos.

[1] <https://github.com/TerraME/terrame/releases/tag/1.6.0>
[2] <https://github.com/TerraME/terrame/releases/tag/2.0-BETA-4>
[3] <https://wiki.qt.io/Transition_from_Qt_4.x_to_Qt5>

3.2 Avaliação de Usabilidade

Avaliar a qualidade de uso é uma parte essencial do processo de desenvolvimento de software. Através dela é possível identificar possíveis problemas enfrentados pelos usuários e melhorias necessárias. Os métodos de avaliação foram usados ao longo do processo de desenvolvimento. Após a refatoração, o próximo passo realizado foi avaliar a qualidade de uso da versão mais recente do DengueME, disponível no início desse trabalho.

Inicialmente foi realizada uma avaliação informal ("rápida e suja") para coleta rápida de *feedback*, por meio da observação de usuários interagindo com o software sendo avaliado. As observações foram realizadas com a participação de alguns membros do Laboratório de Engenharia e Desenvolvimento de Sistemas (LEDS) do Instituto de Ciências Exatas e Aplicadas (ICEA) da Universidade Federal de Ouro Preto (UFOP). Cada participante recebeu um roteiro com tarefas a serem realizadas usando o DengueME. A observação da interação com o sistema foi feita de forma individual, e não havia um tempo máximo para que as tarefas fossem completadas. A coleta de dados ocorreu por meio de anotações, realizadas pelo avaliador durante a realização da observação, e ao final algumas perguntas eram feitas aos participantes.

Posteriormente, foi realizada uma avaliação heurística. O *framework* DECIDE (BARBOSA; SILVA, 2010b) foi utilizado para planejar a avaliação, que teve como objetivo avaliar a usabilidade das funcionalidades principais do DengueME - criação e parametrização de modelos (módulo *Editor*) e criação de interfaces para modelos (módulo *Model Builder*). As interfaces foram inspecionadas por meio de um conjunto de tarefas, que cobriam as funcionalidades dos módulos mencionados, e heurísticas de usabilidade (NIELSEN; MOLICH, 1990). Maiores detalhes sobre a avaliação, motivação e objetivos, planejamento, realização, resultados e análises podem ser encontrados no Apêndice A.

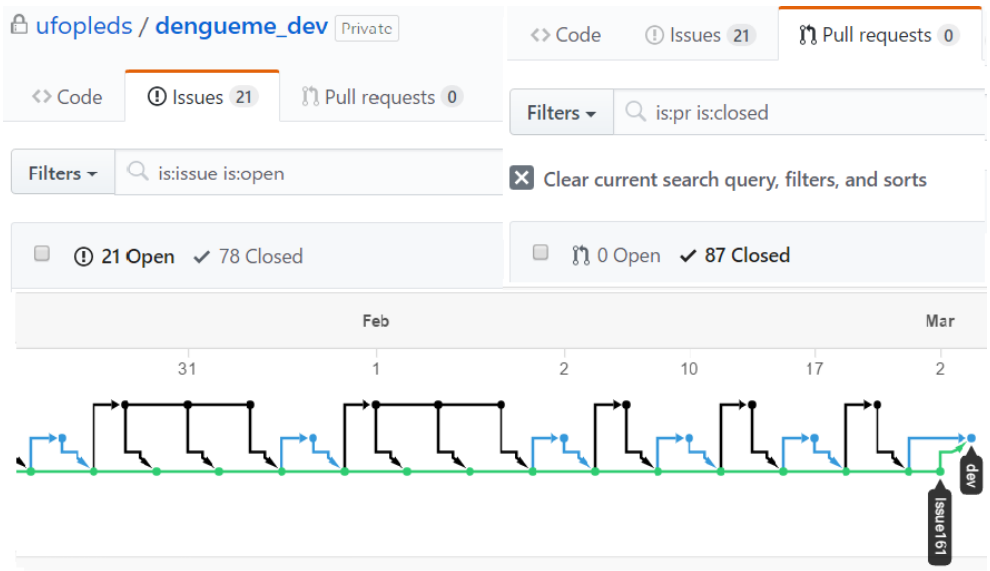
3.3 Desenvolvimento de Software

Métodos de desenvolvimento ágil, se bem aplicados, podem agregar grande valor ao produto final. Alguns princípios e ferramentas foram utilizadas para auxiliar no desenvolvimento do software. Por exemplo, ciclos curtos de desenvolvimento semelhantes aos *sprints* do Scrum[4] foram definidos e usados. Cada ciclo compreendia um conjunto de melhorias e/ou problemas a serem resolvidos (semelhante ao *Sprint Backlog*, do Scrum). Ao final de cada ciclo, aquilo que havia sido desenvolvido era avaliado e validado, para a geração de uma nova versão do software. As atividades a serem realizadas em cada ciclo eram definidas com base em prioridades, visando gerar valor para usuários e atingir os objetivos específicos desse trabalho.

[4] <https://www.scrumalliance.org/why-scrum/scrum-guide>

O serviço GitHub (GITHUB, 2017) foi utilizado como sistema de controle de versão e também para controle de mudanças. Ele oferece diversos recursos, tais como: permite criar tarefas denominadas *issues*, definir e atribuir rótulos para tarefas (ex. prioridade, bug, melhoria, tarefa), definir e atribuir datas-limite *milestones* para as tarefas, dentre outros. As *milestones* foram utilizadas para definir os ciclos de desenvolvimento, ou seja, as *issues* eram incluidas em um *milestone* específico, que indicava a data limite em que as tarefas referentes àquele ciclo deveriam estar prontas. Para organizar o repositório e versionamento optou-se por utilizar o *Branch Workflow* (ATLASSIAN, 2017). Para cada *issue* um *branch* diferente era criado. Modificações eram feitas nesse *branch*, e após as modificações serem verificadas e testadas, o código-fonte do *branch* era combinado de volta com o código original (operação *merge*). Para ilustrar, é possível visualizar na Figura 4 a estrutura de *branchs* e *merges* a partir da criação e resolução de *issues*, que basicamente envolve realizar no repositório as operações de *clone*, *commit* e *pull request*. Até a data da entrega desse documento, haviam sido resolvidos 78 *issues*, realizados 184 *commits* e 87 operações de *pull request*.

Figura 4 – Gráfico do *branch workflow* no repositório do DengueME: criação de novos *branchs* para resolução de *issues* e operações de *pull request* e merge, para atualização das modificações



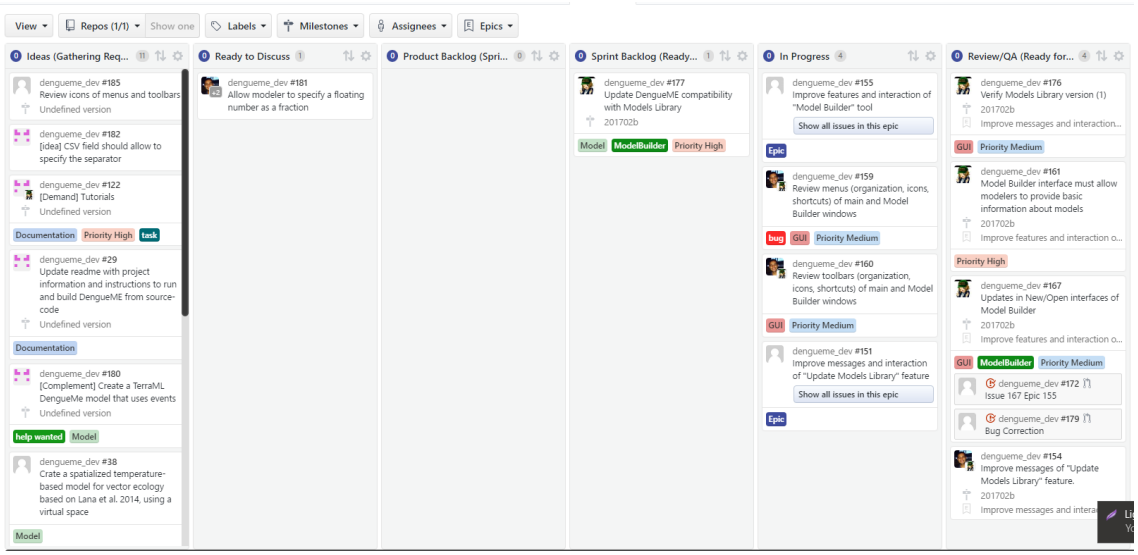
Fonte: Repositório do DengueME no GitHub <https://github.com/ufopleds/dengueme_dev/>

A ferramenta ZenHub Board (ZENHUB, 2017) foi utilizada para auxiliar a gerenciar as atividades do projeto. Ela consiste em um *plugin* para navegadores web que permite visualizar as *issues* de um repositório do GitHub no formato de um quadro KanBan. Ela também oferece outros recursos de apoio ao gerenciamento de projetos, não suportados nativamente pelo GitHub, como por exemplo permite estimar o esforço de *issues* e relacionar

issues como se fossem estórias de usuários e épicos, e visualizar o desempenho por meio de gráficos.

Um *workflow* foi definido e utilizado para realização das atividades do projeto. Ele define um conjunto de etapas sequenciais pelas quais uma *issue* deve passar, ou seja, um *pipeline*, que pode ser mapeado em um Kanban. A representação desse *workflow* em colunas de um quadro Kanban é facilitada pelo ZenHub, como ilustrado na Figura 5.

Figura 5 – Visualização do *workflow* através de um quadro Kanban



Fonte: Repositório do DengueME no GitHub <https://github.com/ufopleds/dengueme_dev/>

O *workflow* é constituído por sete passos, que foram mapeados em colunas do Kanban com o uso do ZenHub. Dessa forma, dentro do *workflow*, as *issues* podem passar por cada uma das etapas, conforme definidas a seguir:

1. *Idea*: inicialmente todas as ideias / demandas do projeto são colocadas nessa coluna, ou seja, toda *issue* é criada a partir daqui. Nesse estágio, a *issue* deve descrever brevemente uma ideia, que pode ser uma nova funcionalidade ou alguma manutenção no software, por exemplo.

2. *Ready to discuss*: quando uma *issue* possuir informação suficiente que permita sua discussão entre a equipe, ela é movida para essa coluna. Nesse ponto, os membros da equipe devem discutir sobre a ideia e ajudar a melhorar sua especificação, com detalhes suficientes que permitam sua implementação futura.

3. *Product backlog*: quando a *issue* possui informações suficientes que permitam definir sua prioridade em relação às demais e discutir sobre sua implementação significa que ela foi adequadamente especificada. Ela então é movida para essa coluna, e

portanto, passa a fazer parte do Product Backlog (semelhante ao Scrum). Modificações ainda podem ocorrer, mas nesse ponto a especificação deve conter informações suficientes que permitam à equipe/cliente priorizar as *issues*, dentre as várias existentes, e planejar a implementação delas. Nesse ponto, as *issues* estão ordenadas por prioridade.

4. *Sprint Backlog*: um grupo de *issues* com maior prioridade é escolhido para ser desenvolvido e entregue ao final de um ciclo (semelhante ao *Sprint Backlog* do *Scrum*). Ou seja, se uma *issue* está nessa coluna, ela deve ser resolvida até o *deadline* correspondente ao final desse ciclo.

5. *In progress*: quando uma *issue* é movida para essa coluna, significa que ela está atribuída a um membro da equipe, e que ele está efetivamente trabalhando para que ela seja resolvida até o final do ciclo atual.

6. *Review*: quando uma *issue* se encontra nesse estágio, significa que ela foi desenvolvida e está pronta para ser testada por outros membros, e posteriormente, avaliada e aprovada pelo cliente. Se algum problema ocorrer, como não passar nos testes ou ser reprovada pelo cliente, a *issue* volta para o *Sprint Backlog* para que os problemas sejam resolvidos.

7. *Closed*: após passar por todas essas etapas, a *issue* é fechada e o produto recebe algum tipo de melhoria (ex. nova funcionalidade ou correção de *bug*). Uma *issue* também pode ser fechada sem passar por todas as etapas, por exemplo, se ela deixar de interessar ao cliente devido à mudanças no escopo do projeto.

3.4 Documentação de software

Durante o desenvolvimento de sistemas de software, uma quantidade significativa de documentos pode ser gerada. Para que sejam úteis, estes documentos devem atender a alguns requisitos básicos (SOMMERVILLE, 2001): servir como forma de comunicação entre os membros da equipe; servir como um repositório de informações para eventuais manutenções; conter informações para ajudar no planejamento do sistema, orçamento de funcionalidades e criação de cronogramas; conter informações relevantes para o usuário, sobre como usar o sistema e onde conseguir ajuda.

Nesse projeto optou-se por não elaborar uma extensiva lista de documentos e artefatos comuns aos processos de engenharia de software. Entretanto, a documentação de usuário e de desenvolvedor são de extrema importância, e deveriam fazer parte de todo produto de software. Em particular, oferecer uma documentação clara, objetiva e completa, mas que não seja demasiadamente extensiva, é um requisito desse projeto, que possui como público-alvo modeladores, pesquisadores e estudantes, muitas vezes sem conhecimentos

avançados de programação. Outro ponto é que o DengueME visa ser um framework *open source* e colaborativo, ou seja, é essencial prover uma documentação adequada para futuros usuários e desenvolvedores colaboradores.

A documentação para desenvolvedores pode ser feita de diversas formas, não necessariamente sendo obrigatório a produção de vários documento adicionais. Por exemplo, é possível documentar o próprio código, e também o projeto por meio de comentários no código e em *commits* realizados. Escrever código bem documentado, de forma clara e compreensível, visando sua manutenabilidade, é também um exemplo de como favorecer o trabalho de futuros desenvolvedores, para que tenham disponíveis toda a informação necessária para o desenvolvimento de novas funcionalidades e correção de *bugs*.

Do ponto de vista dos usuários, um sistema de ajuda interno ao próprio software é uma forma de documentar a ferramenta e apoiar sua utilização, favorecendo o aprendizado de uso. Além disso, documentos externos com orientações de uso, exemplos e tutoriais também são importantes e contribuem para melhorar a experiência de uso. Para essa finalidade, a ferramenta *wiki* do GitHub foi utilizada para elaboração de documentos com orientações de uso e de desenvolvimento. São exemplos de informações disponíveis nessa documentação: descrição da ferramenta, orientações de instalação e compilação da ferramenta, instruções de uso para criação de novos modelos e simulação daqueles já oferecidos pela ferramenta.

4 Resultados

Neste capitulo são apresentados os resultados obtidos pela realização desse trabalho, que incluem refatoração do código-fonte, avaliação de usabilidade, desenvolvimento de melhorias e novas funcionalidades, elaboração de documentação e distribuição do software.

4.1 Refatoração

A refatoração do código-fonte contribuiu para melhorar a legibilidade e manutenabilidade do código-fonte. Diversas modificações foram feitas, tais como: remoção de código obsoleto/desnecessário, organização do código, inserção e adequação de comentários, melhoria dos rótulos usados como identificadores de variáveis e funções.

Os arquivos *XML* e *Lua* gerados pelo DengueME também sofreram alterações em sua organização e conteúdo. O *Model Builder* gera um arquivo XML corresponde à interface gráfica de um modelo, descrevendo os elementos que ela contém, sua organização e valores *default*. Um arquivo Lua, contendo os valores dos parâmetros de entrada, execução e saída é gerado pelo DengueME para que seja possível executar a simulação através de uma chamada à plataforma TerraME.

A etapa de refatoração também incluiu a atualização do *framework* Qt, da versão 4.8 para a versão 5.7. A lista de mudanças entre essas versões pode ser encontrada na documentação oficial do Qt[1]. Além disso, os modelos disponíveis, desenvolvidos em TerraME, foram atualizados da versão 1.6 [2] para a versão 2.0-beta-4[3]. Essas atualizações demandaram algumas mudanças no código-fonte do DengueME e também nos arquivos gerados pela ferramenta.

4.2 Avaliação de Usabilidade

A avaliação de usabilidade do DengueME, que envolveu avaliação heurística e avaliação "rápida e suja", permitiu identificar aspectos negativos e positivos sobre a qualidade de uso da interface. Esse tipo de avaliação é importante pois ajuda a encontrar potenciais dificuldades enfrentadas pelos usuários durante a interação. Por exemplo, um problema da versão inicial, capturado pela avaliação, ocorria durante a parametrização de um modelo quando o usuário selecionava a opção *"Default"*, que redefine os valores dos parâmetros para seu estado original. A ferramenta se comportava de modo inapropriado,

[1] <https://wiki.qt.io/Transition_from_Qt_4.x_to_Qt5>
[2] <https://github.com/TerraME/terrame/releases/tag/1.6.0>
[3] <https://github.com/TerraME/terrame/releases/tag/2.0-BETA-4>

redefinindo os valores conforme esperado, mas fechando o modelo que estava aberto e sendo editado pelo usuário. Dessa forma, o usuário era obrigado a abrir novamente o modelo para então perceber que os valores haviam sido modificados para seu valor original.

A avaliação "rápida e suja"através da observação da interação de usuários com o sistema ocorreu com a participação de membros do Laboratório LEDS, estudantes de computação, ou seja, pessoas com conhecimentos prévios em programação, mas que não são necessariamente público-alvo da ferramenta. Ela teve como principal objetivo analisar a forma como os usuários iriam interagir com a ferramenta, sem nenhum tipo de orientação ou treinamento inicial. Para isso, os participantes foram convidados a realizar tarefas simples envolvendo dois dos principais cenários de uso: 1) criar, parametrizar e simular um modelo; 2) criar a interface gráfica para um novo modelo. A avaliação permitiu identificar diversos problemas de design, como por exemplo: tamanho dos ícones inadequado; problemas com a estrutura e ordenação de menus; mensagens confusas; inexistência de um sistema de ajuda; inconsistência de menus e atalhos. Após a análise dos resultados da avaliação "rápida e suja"com a observação de uso, o DengueME foi atualizado para correção dos problemas encontrados. Diversas mudanças foram feitas na interface e uma nova versão do software foi produzida e novamente avaliada, para verificar a adequação das correções realizadas.

Posteriormente à avaliação "rápida e suja"e às melhorias implementadas, foi conduzida uma avaliação heurística para identificar outros problemas ainda existentes. Essa avaliação permitiu identificar problemas de interface/interação relacionados à usabilidade, como por exemplo sobre a visibilidade do status do sistema, consistência, e prevenção de erros. Uma descrição detalhada dessa avaliação pode ser encontrada no apêndice A.

A figura 6 ilustra um problema de usabilidade, relacionado à inconsistência interna. Na versão anterior do software (Figura 6, imagem à esquerda), a barra de ferramentas era dinâmica e mudava sua aparência de acordo com a funcionalidade que estava sendo utilizada. Embora esse comportamento dinâmico possa ser interessante e aplicado em algumas situações, ele era inconsistente com o comportamento geral da ferramenta, que apresenta menus, ícones e barras de ferramenta estáticos. Esse comportamento dinâmico e a inconsistência podem confundir os usuários. Optou-se por fixar todos ícones na barra de ferramentas, deixando-os desabilitados quando seu uso não fosse permitido dentro de um determinado contexto. A versão atualizada é apresentada na Figura 6, imagem à direita. É possível visualizar ainda a melhoria em relação aos ícones e layout da interface.

Outros problemas de usabilidade que foram resolvidos são ilustrados na figura 7. Na versão anterior, imagem à esquerda, os ícones e a fonte dos menus da interface principal eram muito pequenos e não eram auto-explicativos (*affordance*). Por exemplo, durante a interação os usuários tiveram grande dificuldade em perceber o botão de *help*. O resultado das melhorias pode ser visualizado na imagem à direita, com a barra de ferramentas reorganizada e a substituição dos ícones.

Figura 6 – Interface do Model Builder: à esquerda, versão antiga, com problemas de usabilidade; à direita, versão atualizada, com melhorias de usabilidade

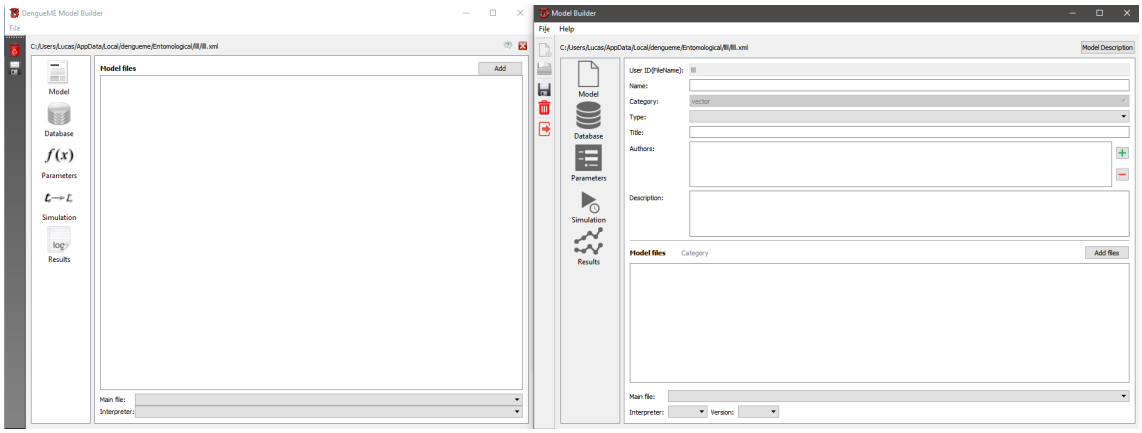
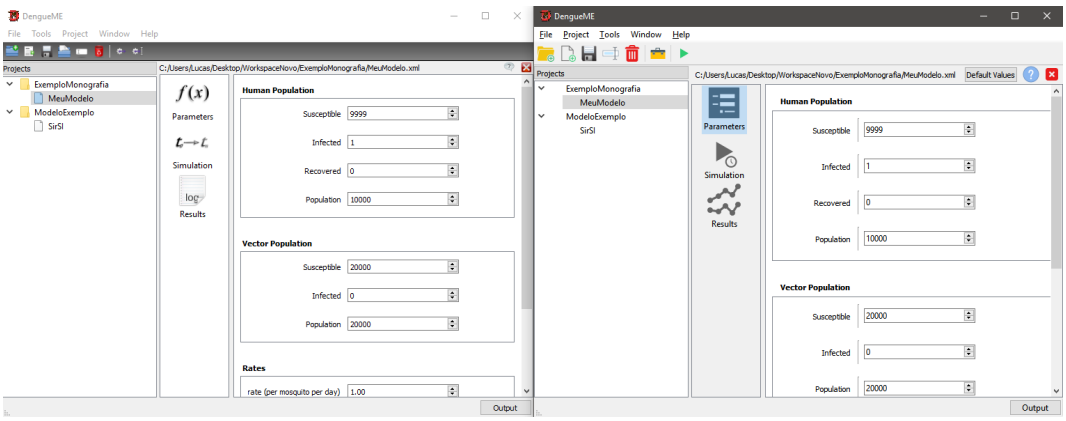


Figura 7 – Interface Principal: à esquerda, versão antiga; à direita, versão com melhorias de usabilidade



4.3 Desenvolvimento

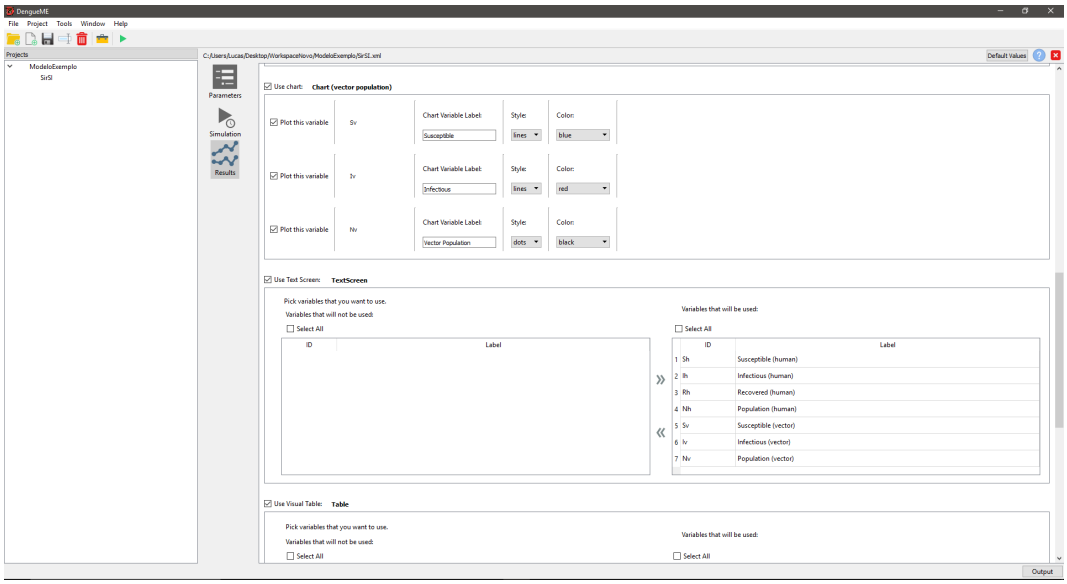
Tendo em vista o objetivo do trabalho de desenvolver melhorias com o intuito de disponibilizar a versão 1.0 do DengueME, os resultados de desenvolvimento incluem correção de *bugs*, melhorias de interface e implementação de novas funcionalidades. Alguns deles foram brevemente apresentados na seção anterior, enquanto outros são apresentados a seguir.

4.3.1 Janela Principal

A janela principal do DengueME é exibida sempre quando o software é inicializado. Sua interface gráfica é constituída por um menu principal, uma barra de ferramentas, um conjunto de *views* e um editor. Por exemplo, a *view Project Explorer* permite visualizar e navegar pelos projetos/modelos por meio de uma árvore de diretórios. O *editor* exibe o modelo para configuração dos parâmetros e execução da simulação. A *toolbar* oferece

acesso rápido às principais ações que o usuário pode executar. A atual versão da interface gráfica da janela principal do DengueME é ilustrada na figura 8.

Figura 8 – Janela principal do DengueME



A partir da interface principal, o usuário pode realizar ações como criar um novo projeto, criar um novo modelo, salvar/remover modelos, configurar a ferramenta. A maior parte das ações possui atalhos rápidos por meio da barra de ferramentas e também do teclado. Quando um usuário cria um novo modelo, a interface correspondente a esse modelo é apresentada. O layout da interface de parametrização do modelo é organizada em abas, de forma consistente com a interface do *Model Builder* (que é usado para criar interfaces para os modelos). Em cada aba o usuário pode definir parâmetros correspondentes a diferentes aspectos do modelo, tais como entrada (*input*), simulação e resultados (*output*).

4.3.2 Model Builder

O *Model Builder* é um módulo que permite criar interfaces gráficas para parametrização de modelos. Dessa forma, ele permite a usuários mais avançados, modeladores, inserir novos modelos na biblioteca e definir a interface gráfica para utilização desses modelos. Ou seja, o *Model Builder* permite ao modelador especificar os elementos de interface gráfica que serão apresentados para os usuários do seu modelo, que permitirão sua parametrização e simulação.

A interface do *Model Builder* é organizada em abas, sendo cada uma delas correspondente a um conjunto de informações específicas que podem ser definidas para um modelo, conforme apresentado a seguir:

- Model: nesta aba o modelador deve inserir informações básicas sobre o modelo, como

por exemplo linguagem/plataforma de simulação, versão e autores. Além dessas informações básicas, todos os arquivos necessários para a execução do modelo devem ser carregados.

- Database: nesta aba o modelador pode prover informações relacionadas à bases de dados utilizadas pelo modelo, se for o caso. Por exemplo, para a plataforma TerraME é possível utilizar bases de dados em MySql, Access, ou mesmo memória virtual (grade de células). Prover tais informações é opcional, dado que nem todos os modelos fazem uso de banco de dados.

- Parameters: nessa aba o modelador pode definir todas as variáveis (parâmetros) que serão exibidas e poderão ser modificadas pelos usuários. É possível organizar os parâmetros em grupos, e também definir informações sobre os parâmetros tais como tipo de dado, valor padrão, valores mínimo/máximo, regras de validação. Todas as variáveis que se deseja fazer visíveis aos usuários devem ser especificadas nessa aba, e o código-fonte correspondente será gerado automaticamente pelo DengueME.

- Simulation: informações sobre a execução do modelo, tais como tempo inicial, tempo final e passos de simulação, devem ser especificados nessa aba.

- Results: informações sobre os resultados (*output*) do modelo, sobre como eles serão exibidos e/ou armazenados devem ser especificados nessa aba. Por exemplo, para o TerraME é possível usar diversos tipos de "visualizadores", conforme o tipo da variável sendo "observada", tais como Log, Chart, TextScreen e VisualTable.

Durante a etapa de refatoração, o *Model Builder* sofreu diversas mudanças em sua interface gráfica e sua arquitetura interna. Parte delas foi necessária devido às mudanças na versão do TerraME, enquanto outras mudanças foram motivadas pela necessidade de novas funcionalidades. Por exemplo, inicialmente a interface gerada pelo Model Builder permitia apenas que o usuário escolhesse se desejava visualizar ou não os resultados da simulação. Com o intuito de dar maior flexibilidade para os usuários, a estrutura do Model Builder foi modificada, permitindo ao usuário selecionar individualmente quais resultados deseja visualizar/salvar (ex. gráficos, mapas, logs), e também realizar alguma customização (ex. legenda, cores). Dessa forma, é possível ao modelador prover maior flexibilidade sobre o uso de seus modelos, dando ao usuário final maior controle sobre a quais resultados ele deseja visualizar/salvar e como.

Outro exemplo de mudança significativa ocorreu na aba *Model*. Inicialmente, ela somente permitia ao modelador carregar os arquivos fonte necessários para a execução do modelo. Na versão desenvolvida por esse trabalho, é possível para o modelador inserir um conjunto de informações relevantes sobre o modelo, tais como descrição do modelo, plataforma de modelagem, versão e autores. Além disso, o modelador pode incluir juntamente

com o modelo uma breve documentação sobre o mesmo. Outra melhoria relacionada à documentação dos modelos, foi a inserção de campos específicos para permitir ao modelador documentar as variáveis, na aba *Parameters*. Além disso, visando melhorar a eficiência de uso, foi implementada a funcionalidade para duplicar (clonar) variáveis.

Dessa forma, por meio do *Model Builder*, o modelador deve carregar todos os arquivos necessários para executar o modelo, fornecer informações que permitam ao usuário configurar o banco de dados do modelo (quando necessário), definir os campos para parametrização do modelo e da simulação, e especificar os resultados gerados pelo modelo e como eles poderão ser apresentados e customizados pelos usuários. Todas essas informações de configuração são armazenadas em um arquivo XML, que descreve a interface gráfica do modelo que está sendo criado. E esse arquivo XML é carregado pelo DengueME, quando o usuário cria um novo modelo daquele tipo, na janela principal.

4.3.3 Biblioteca de modelos

Um dos requisitos do DengueME é que ele seja extensível, ou seja, o *framework* deve facilitar a disponibilização, inclusão, compartilhamento e organização de modelos sobre a dengue. Nesse sentido, o DengueME oferece uma biblioteca com alguns modelos básicos sobre a dinâmica de transmissão da dengue e também sobre a ecologia do vetor. Entretanto, na versão anterior, essa biblioteca era completamente acoplada ao software, ou seja, para atualizar a biblioteca de modelos era necessário atualizar o software. Portanto, a inclusão de novos modelos, ou mesmo alteração daqueles já existentes, exigia a geração de uma nova versão do software.

Uma grande melhoria produzida por esse trabalho foi promover o desacoplamento entre o DengueME e a sua biblioteca de modelos, viabilizando a realização de atualizações de forma independente. Para isso, foi necessário disponibilizar publicamente a biblioteca de modelos por meio de um servidor remoto. Um repositório público no GitHub está sendo utilizado como servidor remoto da biblioteca de modelos. Dessa forma, a biblioteca do DengueME é organizada em dois componentes:

- Biblioteca Global: é a biblioteca "oficial"do DengueME, que contém todos os modelos disponibilizados pelo DengueME. Visando facilitar a colaboração e compartilhamento de modelos, a biblioteca está hospedada em um repositório público no GitHub[4]. Atualmente, ainda não é possível solicitar a inclusão de modelos na biblioteca global por meio da interface gráfica do DengueME. Isso será feito futuramente. Entretanto, a interface do GitHub já permite e facilita esse tipo de colaboração. Por exemplo, caso um modelador queira acrescentar um modelo à biblioteca global do DengueME, ele poderá clonar o repositório (*fork*) e enviar atualizações com a inclusão do novo

[4] <https://github.com/ufopleds/DengueMELib>

modelo por meio de um *pull request*. As mudanças poderão ser verificadas e testadas pelos colaboradores oficiais, e poderão ser incluídas e disponibilizadas para todos os usuários em futuras versões da biblioteca de modelos (sem a necessidade de atualizar o software DengueME).

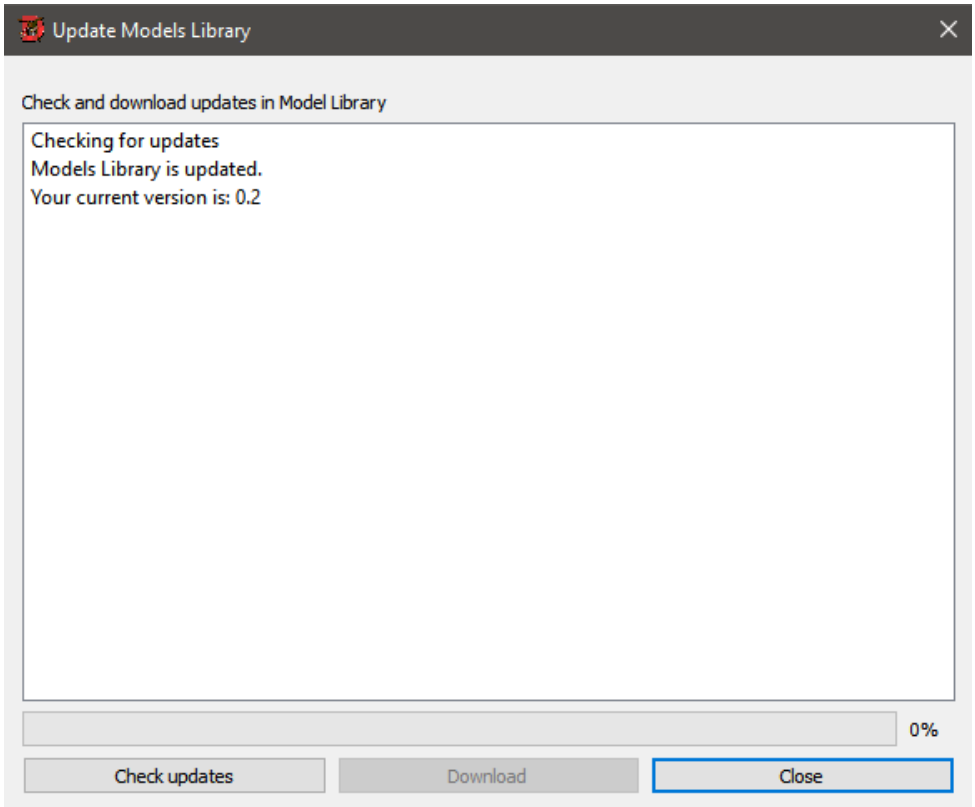
- Biblioteca Local: localmente, no computador do usuário, a biblioteca é composta por uma cópia local da biblioteca global e pela biblioteca pessoal do usuário. A cópia local da biblioteca global se faz necessária para permitir o uso do software sem a necessidade de conexão com a internet. Ela está localizada no diretório *Models*, contido no diretório raiz do DengueME. Por sua vez, a biblioteca pessoal do usuário contém os modelos que foram criados pelo usuário com o uso do *Model Builder*. Ela se localiza no diretório de dados de aplicação do usuário, cujo caminho depende do sistema operacional e configurações utilizadas.

Dada essa organização com versões local e global da biblioteca de modelos do DengueME, se faz necessário oferecer recursos que permitam a sincronização entre elas. No momento, somente é possível ao usuário realizar atualizações na biblioteca oficial de modelos. Futuramente, espera-se que também seja possível aos usuários o envio e compartilhamento de sua biblioteca pessoal de modelos, através de repositórios remotos. Portanto, foi necessário oferecer funcionalidades que permitissem atualizar a biblioteca local de modelos, conforme a última versão disponível na biblioteca global.

Atualmente não é realizada uma varredura automática para verificar se há versões mais novas disponíveis na biblioteca global. Para isso, o usuário deve abrir o módulo de atualização e verificar se existem versões mais novas disponíveis. A atual versão da biblioteca local é armazenada em um arquivo XML, e é comparada à versão da biblioteca global, disponível no repositório. Caso a versão global seja mais nova que a versão local, é oferecida ao usuário a opção de fazer o download do arquivo com a nova versão da biblioteca. Feito isso, o usuário deve extrair os arquivos para o diretório *Models*, localizado dentro da pasta onde o DengueME foi instalado. As seguintes condições são verificadas para permitir o download do arquivo: a pasta *Models* não foi encontrada dentro do diretório raiz do DengueME; o arquivo de configuração da biblioteca não foi localizado dentro da pasta *Models*; o arquivo de configuração indica que a versão local da biblioteca de modelos está desatualizada em relação à versão global.

A figura 9 ilustra a interface de atualização da biblioteca de modelos, através da qual é possível verificar se há novas atualizações e realizar o *download* das atualizações disponíveis.

Figura 9 – Interface do updater da biblioteca de modelos



4.4 Documentação de software

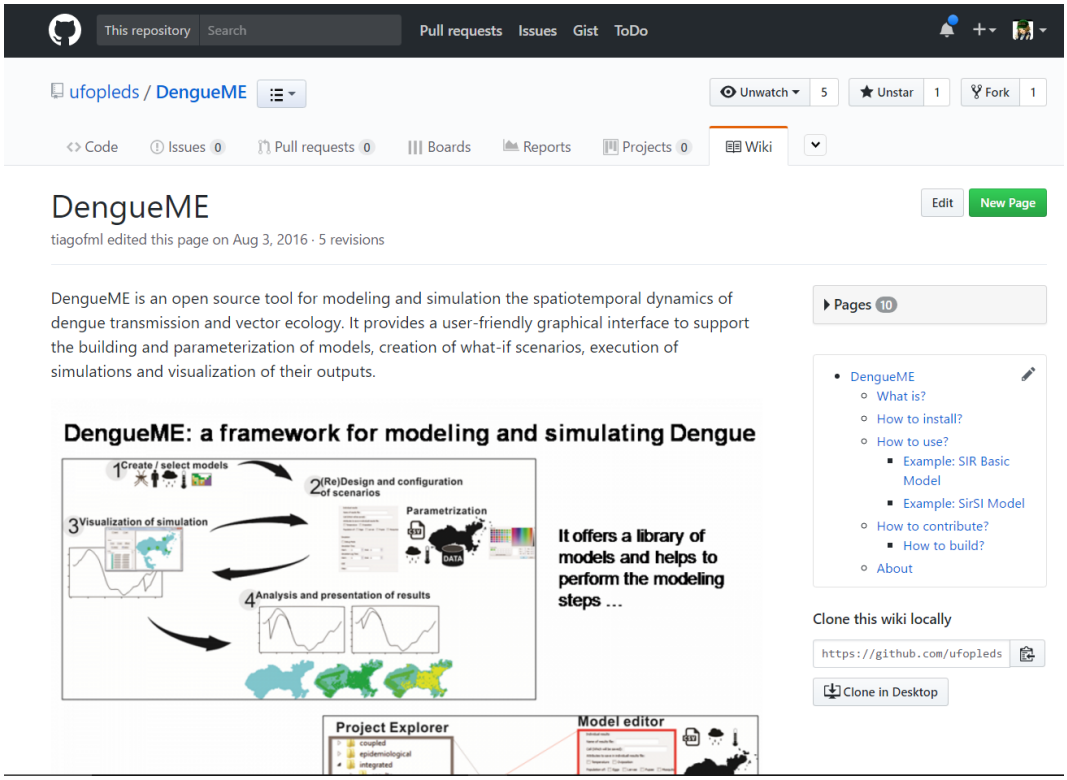
Uma das contribuições do trabalho foram as melhorias em relação à documentação de software, para desenvolvedores e usuários. Do ponto de vista de desenvolvedores, a etapa de refatoração incluiu a documentação do código-fonte por meio da inclusão de comentários, melhorias de legibilidade e compreensão do código (ex. uso de nomes significativos). Outro ponto foi a documentação do desenvolvimento, incluindo decisões de projeto e especificação de requisitos, diretamente no repositório, através das *issues* do GitHub. Dessa forma, é possível por exemplo rastrear as mudanças que ocorreram e também compreender as decisões de design que foram tomadas. Portanto, o repositório em si, graças às funcionalidades de versionamento e controle de mudanças oferecidas pelo GitHub, constitui uma importante fonte de informação e documentação do projeto, tanto para desenvolvedores quanto para usuários.

Sob o ponto de vista dos usuários, uma importante melhoria foi o desenvolvimento do sistema de ajuda interna do software, que oferece informações básicas sobre o uso da interface e dos modelos. Além disso, foi elaborada e disponibilizada na wiki do repositório público do DengueME[5], uma documentação de usuário (Figura 10). Ela contém informações básicas sobre a ferramenta e seu funcionamento, instruções sobre como utilizar, quais as

[5] <https://github.com/ufopleds/DengueME/wiki/DengueME>

dependências para execução do fonte, informações sobre como contribuir com o projeto ou contatar os envolvidos no projeto, e exemplos de como usar o software.

Figura 10 – Documentação de usuário - página inicial



Fonte: Repositório do DengueME no GitHub <https://github.com/ufopleds/dengueme/>

4.5 DengueME 1.0

A realização desse trabalho contribuiu para resolver um conjunto de problemas e implementar uma série de melhorias. Algumas delas foram apresentadas sem seções anteriores, tais como atualização de dependências de software, melhorias de usabilidade, elaboração de documentação, disponibilização da biblioteca de modelos em servidor remoto. Outra contribuição foi a disponibilização da ferramenta em dois idiomas: inglês e português.

O DengueME está disponível publicamente em sua versão 1.0, juntamente com o seu código-fonte, no link <https://github.com/ufopleds/DengueME>. O código-fonte está sendo distribuído sob a licença *BSD - 2 clause*, uma licença *open source* que permite a modificação e redistribuição do código por qualquer pessoa, desde que os créditos sejam dados aos autores, sendo estes os desenvolvedores originais e qualquer outro que tenha modificado o código após a distribuição do fonte. O texto base da licença pode ser encontrado no seguinte link: <https://opensource.org/licenses/BSD-2-Clause>.

O DengueME utiliza a biblioteca Qt em sua licença *open source*, que possui como requisito a disponibilização da ferramenta juntamente dos arquivos *dll* do Qt. Ou seja, essa

licença não permite a geração de executáveis com *link* estático para as dlls (dlls localizadas dentro do .exe). Atualmente, os modelos disponíveis no DengueME estão implementados na plataforma TerraME, versão 2.0-beta-4. Portanto, para executá-los, é necessário instalar a referida versão do TerraME.

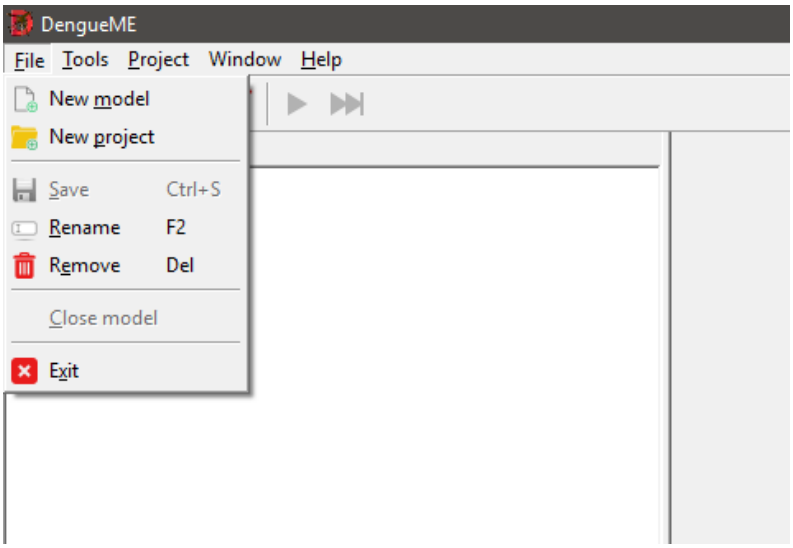
O DengueME 1.0 foi testado nos ambientes Windows 10/8/7 e Linux (distribuições Ubuntu 14-16, Mint 17, OpenSuse, Manjaro 5.7.5, Debian 8). Em todas elas o software apresentou o funcionamento desejado. A única limitação nos testes, quando não foi possível executar os modelos, foi devido à plataforma TerraME, que não oferece versão testada para a grande maioria das distribuições Linux. É possível executar o DengueME através códifo-fonte, disponível no repositório. Para isso, é necessário instalar o *framework* QT 5.7 e Qt Creator. Detalhes sobre a instalação do DengueME e uso a partir do código-fonte podem ser encontrados na documentação disponível no repositório.

4.5.1 Exemplo de uso

Para ilustrar o funcionamento do DengueME, apresentamos nessa subseção uma demonstração de uso do software que inclui a criação e parametrização de um modelo disponível na biblioteca, sua simulação e visualização dos resultados. O modelo utilizado é o SIR-SI, um modelo compartimentado de transmissão, que considera as populações de humanos e de vetor, nos estados suceptível (S), infeccioso (I) e recuperado (R).

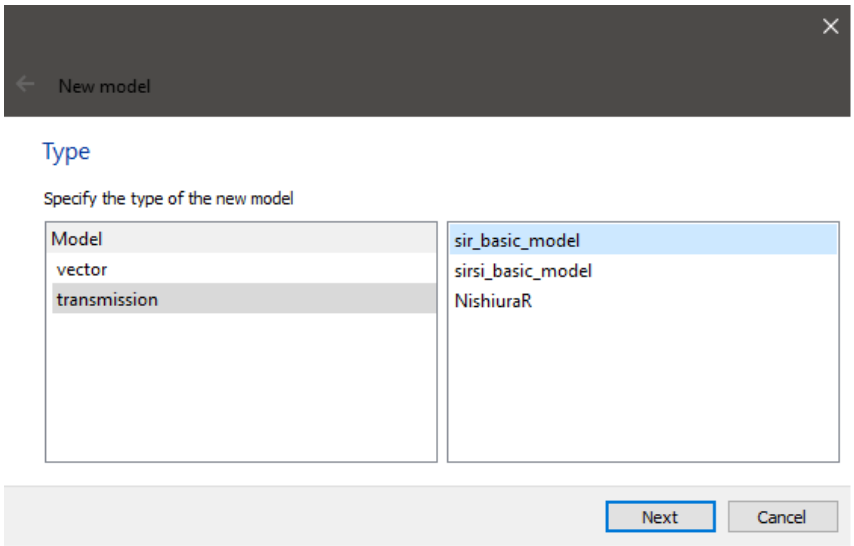
Os seguintes passos podem ser realizados para a criação, parametrização e simulação do modelo. Primeiro, no menu da janela principal do DengueME, selecione a opção File -> New Project (Figura 11). Escolha um nome para o projeto e clique em *Finish*. Depois, selecione a opção File -> New Model (Figura 11) no menu da janela principal do DengueME. Um *wizard* irá auxiliar na criação do novo modelo.

Figura 11 – Exemplo de uso - Passo 1: criando um novo projeto. Passo 2: criando um novo modelo.



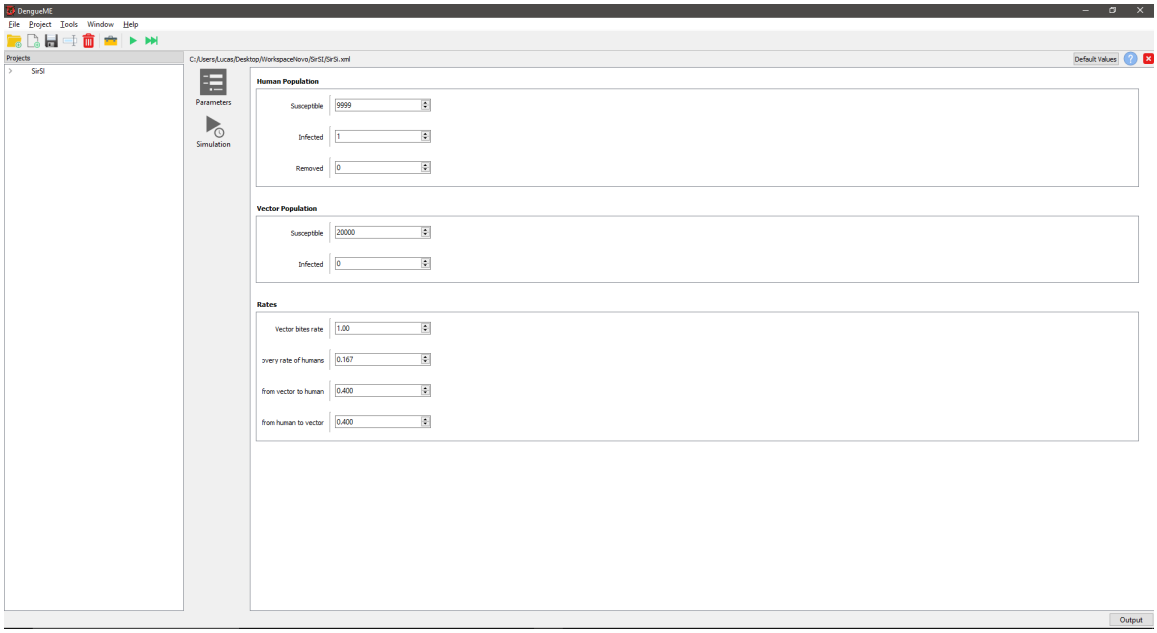
Para criar o modelo, inicialmente é necessário escolher o tipo de modelo que se deseja criar. Para isso, selecione o modelo *sirsi-basic-model*, disponível na categoria *transmission* (Figura 12). Clique em *Next*, selecione o projeto e defina um nome para o modelo que está sendo criado.

Figura 12 – Exemplo de uso - Passo 2: criando um novo modelo



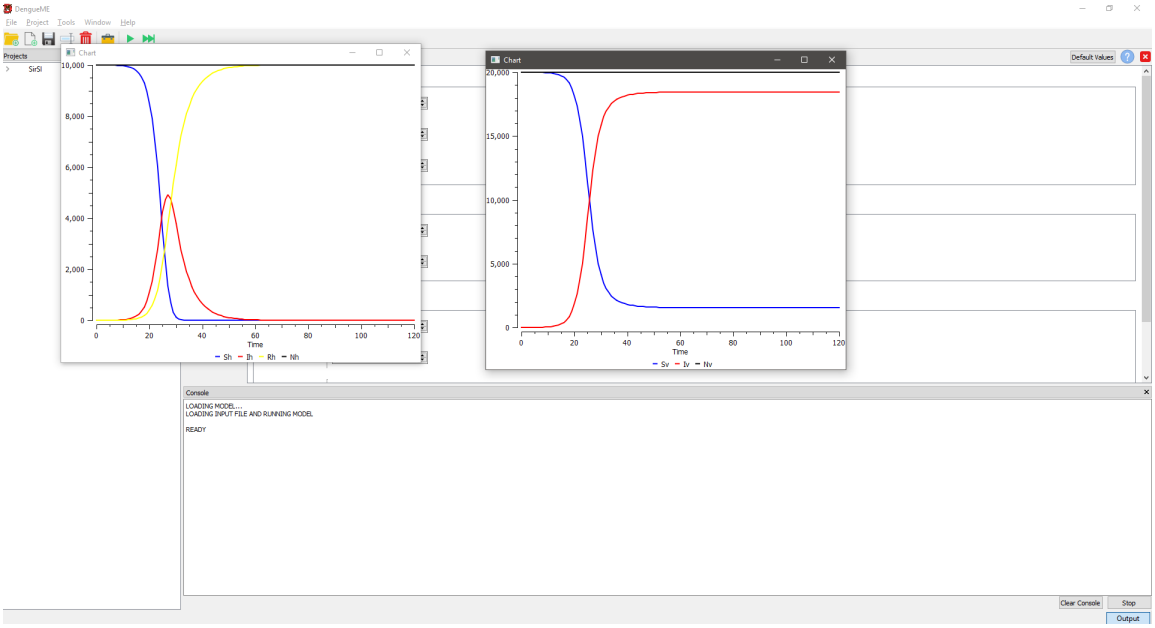
Uma vez criado o modelo, é possível visualizar sua interface de edição. No lado esquerdo da janela principal, na *view Project Explorer*, é possível visualizar os projetos e modelos. E no centro, é possível visualizar o *Editor* (Figura 13), que é usado para configurar o modelo que está aberto atualmente. Após configurar os parâmetros conforme desejado, é possível executar a simulação através do menu principal ou da barra de ferramentas.

Figura 13 – Exemplo de uso - Passo 3: configurando os parâmetros do modelo



A simulação do modelo SIR-SI irá produzir um conjunto de resultados, que incluem dois gráficos com as populações de humanos e de vetor (Figura 14), em cada um dos estágios possíveis (S-I-R, para humanos, e S-I, para mosquitos). Para encerrar a simulação basta fechar as janelas dos gráficos de resultados. Se desejar, poderá repetir o processo alterando o valor dos parâmetros e analisando o impacto das mudanças nos resultados da simulação.

Figura 14 – Exemplo de uso - Passo 4: configurando os parâmetros do modelo



5 Considerações Finais

A dengue é um grande desafio para a saúde pública no Brasil e no mundo. O estudo da dinâmica da Dengue e de seu vetor podem ajudar a mudar esse cenário, pois contribuem para a melhor compreensão do problema e possibilidades de mitigação. A modelagem e simulação computacional é um exemplo de ferramenta que pode ser empregada no estudo de dinâmicas complexas, como por exemplo a dinâmica de transmissão da dengue. Nesse contexto, o DengueME (*Dengue Modeling Environment*) é um *framework* de software que visa apoiar o estudo da dengue através da modelagem e simulação computacional das dinâmicas de transmissão do vírus e de ecologia do vetor. Entretanto, a ferramenta ainda em estágio preliminar apresentava diversos problemas.

Esse trabalho teve como objetivo implementar as melhorais necessárias no software DengueME para disponibilizá-lo publicamente em sua versão 1.0. Para isso, inicialmente foi realizada uma refatoração do seu código-fonte e atualização das dependências de software (Qt e TerraME) para as versões mais recentes disponíveis na época de início desse projeto. Além disso, métodos de avaliação de interface foram empregados com o intuito de corrigir problemas de interface/interação, como por exemplo usabilidade, a fim de melhorar a qualidade de uso do software. O processo de desenvolvimento de software foi inspirado em métodos ágeis, principalmente no Scrum. Para isso, foi definido um *workflow*, e utilizados os recursos oferecidos pelos serviços GitHub e ZenHub para versionamento, controle de mudanças e gerenciamento de projetos. Algumas contribuições significativas foram obtidas, tais como: disponibilização da biblioteca de modelos em um servidor remoto, o que permite atualizações independentes no software e nos modelos; elaboração de documentação de usuário e de desenvolvedor; melhorias na internacionalização do software - disponível em inglês e português e recursos que facilitam sua tradução para outros idiomas; disponibilização da versão 1.0 do software, para ambientes Windows e Linux, em um repositório público no GitHub <https://github.com/ufopleds/DengueME/>.

Desenvolver software não é uma tarefa trivial, principalmente quando o software é voltado para aplicações distintas de nossa área de conhecimento, e destinado a um público-alvo diversificado, com diferentes perfis e formações. Vários desafios e dificuldades foram encontrados e superados ao longo desse trabalho, e muitos outros surgirão na medida em que as pessoas começarem a efetivamente utilizar a ferramenta.

Os trabalhos futuros incluem a avaliação com um grupo de usuários-alvo e oferecer suporte para que modelos sejam implementados na linguagem R. Além disso, recursos para apoiar as atividades de modelagem podem ser incluídos em futuras versões, como por exemplo funcionalidades para análise de sensibilidade de parâmetros, edição colaborativa de

modelos, e integração de modelos. Por fim, para que o DengueME seja realmente utilizado como ferramenta de saúde pública ou mesmo para fins acadêmicos/didáticos, novos modelos precisam ser desenvolvidos e incorporados à ferramenta. E recursos adicionais de apoio aos usuários, tais como tutoriais, vídeos e cursos também devem oferecidos.

Referências

ATLASSIAN. *BranchWorkflow*. 2017. <https://www.atlassian.com/git/tutorials/comparing-workflows#feature-branch-workflow>. Acesso em: 25 fev. 2017. Citado na página 29.

BAIANU, P. D. I. *Complexity, Emergent Systems and Complex Biological Systems: Complex Systems Theory and Biodynamics. [Edited book by I.C. Baianu, with listed contributors (2011)]*. PediaPress: Mainz, Germany, 2011. 1–228 p. In print, December 2011. Disponível em: <http://cogprints.org/7736/>. Citado na página 19.

BARBOSA, S.; SILVA, B. *Interação Humano-Computador*. Elsevier Brasil, 2010. ISBN 9788535211207. Disponível em: <https://books.google.com.br/books?id=qk0skwr_cewC>. Citado 3 vezes nas páginas 51, 53 e 54.

BARBOSA, S.; SILVA, B. *Interação Humano-Computador*. [S.l.]: Elsevier Brasil, 2010. ISBN 9788535211207. Citado 2 vezes nas páginas 24 e 28.

BENYON, D. e. *Designing Interactive Systems: People, Activities, Contexts, Technologies*. Addison-Wesley, 2005. ISBN 9780321116291. Disponível em: <https://books.google.com.br/books?id=iWe7VkFW0zMC>. Citado na página 24.

BROWN, T.; MCLAFFERTY, S.; MOON, G. *A Companion to Health and Medical Geography*. Wiley, 2009. (Wiley Blackwell Companions to Geography). ISBN 9781444314779. Disponível em: <https://books.google.com.br/books?id=xRbvpGJRwzcC>. Citado na página 17.

CARNEIRO, T. G. de S. et al. An extensible toolbox for modeling nature–society interactions. *Environmental Modelling & Software*, Elsevier, v. 46, p. 104–117, 2013. Citado na página 18.

CâMARA, F. et al. Clima e epidemias de dengue no estado do rio de janeiro. *Rev Soc Bras Med Trop*, v. 42, p. 137–40, 2009. Citado na página 17.

ECLIPSE. *Spatiotemporal Epidemiological Modeler*. 2017. <http://www.eclipse.org/stem/>. Acesso em: 8 fev. 2017. Citado na página 21.

EPSTEIN, J. M. Why Model? *Journal of Artificial Societies and Social Simulation*, v. 11, n. 4, 2008. Citado na página 19.

FOWLER, M.; BECK, K. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley, 1999. (Component software series). ISBN 9780201485677. Disponível em: <https://books.google.com.br/books?id=1MsETFPD3I0C>. Citado na página 27.

GITHUB. *GitHub*. 2017. <http://github.com/>. Acesso em: 25 fev. 2017. Citado 2 vezes nas páginas 25 e 29.

GRIFFIN, J.; STOYANOV, D. *MalariaTool*. 2015. <https://www1.imperial.ac.uk/malariamodelling/toolsdata/tools/>. Acesso em: 20 fev. 2017. Citado na página 21.

GRIFFIN, J. T. et al. Reducing plasmodium falciparum malaria transmission in africa: A model-based evaluation of intervention strategies. *PLOS Medicine*, Public Library of Science, v. 7, n. 8, p. 1–17, 08 2010. Disponível em: <http://dx.doi.org/10.1371%2Fjournal.pmed.1000324>. Citado na página 21.

HEWETT, T. et al. *ACM SIGCHI Curricula for Human-computer Interaction*. New York, NY, USA, 1992. Citado na página 24.

HLADISH, T. et al. Epifire: An open source c++ library and application for contact network epidemiology. *BMC Bioinformatics*, v. 13, n. 1, p. 1–12, 2012. ISSN 1471-2105. Disponível em: <http://dx.doi.org/10.1186/1471-2105-13-76>. Citado na página 21.

LIMA, T. et al. A framework for modeling and simulating aedes aegypti and dengue fever dynamics. In: *Proceedings of the 2014 Winter Simulation Conference*. Piscataway, NJ, USA: IEEE Press, 2014. (WSC '14), p. 1481–1492. Disponível em: <httpdl.acm.orgcitation.cfmid=2693848.2694037>. Citado 2 vezes nas páginas 22 e 23.

LIMA, T. F. M. de et al. Dengueme: A tool for the modeling and simulation of dengue spatiotemporal dynamics. *International Journal of Environmental Research and Public Health*, v. 13, n. 9, 2016. Citado 2 vezes nas páginas 17 e 22.

MOCKUPS. *Mockups*. 2017. <https://moqups.com/>. Acesso em: 25 fev. 2017. Citado na página 25.

NIELSEN, J.; MOLICH, R. Heuristic evaluation of user interfaces. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, 1990. (CHI '90), p. 249–256. ISBN 0-201-50932-6. Disponível em: <http://doi.acm.org/10.1145/97243.97281>. Citado 3 vezes nas páginas 28, 51 e 58.

NOGUEIRA, R. M. et al. Dengue in the state of rio de janeiro, brazil, 1986-1998. *Mem Inst Oswaldo Cruz*, v. 94, n. 3, p. 297–304, 1999. Citado na página 17.

QTCOMPANY. *Qt Framework*. 2017. <https://wiki.qt.io/Main>. Acesso em: 3 mar. 2017. Citado na página 18.

SHARP, H.; ROGERS, Y.; PREECE, J. *Interaction Design: Beyond Human-Computer Interaction*. Wiley, 2007. ISBN 9780470018668. Disponível em: <https://books.google.com.br/books?id=kcEZAQAAIAAJ>. Citado 2 vezes nas páginas 51 e 53.

SOKOLOWSKI, J.; BANKS, C. *Modeling and Simulation Fundamentals: Theoretical Underpinnings and Practical Domains*. 1. ed. [S.l.]: Wiley, 2010. Citado 3 vezes nas páginas 17, 19 e 20.

SOMMERVILLE, I. Software documentation. In: *In Software Engineering, vol 2: The supporting Processes. R.H. Thayer and M.I. Christensen (eds), Willey-IEEE*. [S.l.]: Press, 2001. Citado na página 31.

SVS-BRASIL, S. de Vigilância em Saúde Ministério da S. *Boletim Epidemiológico*. 2016. <http://portalsaude.saude.gov.br/images/pdf/2016/janeiro/15/svs2016-be003-dengue-se52.pdf>. Acesso em: 6 jul. 2016. Citado na página 17.

TERRALAB. *TerraME*. 2016. <http://www.terrame.org/doku.php>. Acesso em: 6 jan. 2017. Citado na página 21.

TERRALAB. *TerraME*. 2017. <https://github.com/TerraME/terrame/wiki>. Acesso em: 3 mar. 2017. Citado na página 18.

TISUE, S.; WILENSKY, U. Netlogo: A simple environment for modeling complexity. p. 16–21, 2004. Citado na página 20.

ZENHUB. *ZenHub*. 2017. <https://www.zenhub.com/>. Acesso em: 25 fev. 2017. Citado na página 29.

APÊNDICE A – Avaliação Heurística

Resumo

DengueME (Dengue Modeling Environment) is a platform that aims to support the study of the dengue dynamics using modeling and computer simulation. DengueME aims to help people without great experience in programming to use models. It offers a friendly graphical user interface allowing users to parametrize models and run simulations. Advanced users can use DengueME to create graphical interfaces to theirs models and share this interface with other users. This work aimed to assess the usability of DengueME interface through heuristic evaluation method. This work aimed to assess the usability of DengueME interface through heuristic evaluation method. Ten heuristics were used, which allowed to find several usability issues. Recommendations were made in order to guide improvements in the software interface.

A.1 Introdução

O DengueME é uma ferramenta para apoiar a modelagem e simulação da dinâmica de transmissão da dengue e da ecologia de seu vetor. Um dos objetivos da ferramenta é facilitar o uso de modelos disponíveis em sua biblioteca por meio de interfaces gráficas, para parametrização, simulação e visualização dos resultados. O DengueME também visa facilitar a criação e compartilhamento de novos modelos relacionados a dengue.

Esse trabalho teve como objetivo avaliar a usabilidade do DengueME através do método de avaliação heurística (BARBOSA; SILVA, 2010a; NIELSEN; MOLICH, 1990). O *framework* DECIDE (SHARP; ROGERS; PREECE, 2007) foi utilizado para planejar a avaliação. Foram utilizadas dez heurísticas de usabilidade (NIELSEN; MOLICH, 1990), que permitiram identificar um conjunto de problemas na interface e interação com o software. A versão *beta* do DengueME foi utilizada na avaliação heurística, que foi conduzida em um *notebook* com o sistema operacional Windows 10. Ao todo, foram detectadas e documentadas vinte e duas falhas, com diferentes graus de severidade. Para cada uma delas, foram apresentadas sugestões de solução para guiar o desenvolvimento de futuras melhorias para o software.

A.1.1 O DengueME

O DengueME é um *framework* voltado para o estudo da dengue por meio da modelagem e simulação computacional. Ele é distribuído como uma aplicação *desktop* para

ambientes Linux e Windows, de código-fonte aberto sobre a licença "BSD-2 Clause". O software se encontra em desenvolvimento, tendo sido utilizada para avaliação a versão *beta*, disponível no link <https://github.com/ufopleds/dengueme>.

O DengueME oferece uma interface gráfica, que permite aos usuários criar, parametrizar e executar diferentes tipos de modelos, disponíveis em sua biblioteca. Cada modelo possui sua própria interface de configuração, e o código-fonte a ser executado pela simulação é gerado automaticamente pela ferramenta. Dessa forma, o usuário pode realizar todas as etapas por meio de uma interface gráfica, desde a criação de um modelo (a partir daqueles disponíveis na biblioteca) à visualização dos resultados da simulação.

Além disso, o DengueME também oferece funcionalidades que permitem estender sua biblioteca de modelos, com a criação de novos modelos. Isso é feito através do módulo *Model Builder*, que permite criar interfaces de parametrização para novos modelos. Através dele, o usuário modelador pode definir os parâmetros que poderão ser configurados pelo usuário final, adicionar os arquivos necessários para a execução da simulação, e definir os resultados que serão apresentados aos usuários.

Dessa forma, o DengueME deve oferecer uma interface com boa qualidade de uso para usuários com diferentes perfis - tanto para "usuários finais"dos modelos, que desejam utilizar a interface gráfica para parametrização e simulação, quanto para "usuários modeladores", que desejam criar novos modelos com respectivas interfaces gráficas para sua utilização. O primeiro grupo de usuários não possui necessariamente experiência em programação, e portanto, realizar alterações diretamente no código-fonte pode ser uma tarefa árdua, que exigiria por exemplo domínio da linguagem de programação utilizada no desenvolvimento do modelo. Assim, permitir que modelos sejam completamente utilizados, incluindo sua parametrização, execução e visualização da simulação, a partir de uma interface gráfica amigável, é um requisito desejável e necessário. O segundo grupo, envolve usuários com experiência em modelagem e programação, com interesse na inclusão e compartilhamento de novos modelos. Para esse grupo, é essencial que o sistema oferece funcionalidades que apoiem a criação de novos modelos e de interfaces gráficas que favoreçam o uso e compartilhamento desses modelos.

Oferecer boa qualidade de uso é um requisito de todo sistema interativo. Mas isso se torna ainda mais importante quando se busca atingir usuários com diferentes perfis de uso. E um dos aspectos iniciais, necessários para se oferecer uma boa qualidade de uso em interfaces, é a usabilidade. O método de avaliação heurística foi escolhido por apresentar custo relativamente baixo e principalmente pela indisponibilidade de usuários reais para avaliar a ferramenta. E apesar das limitações do método de avaliação heurística, ele possui uma boa relação custo-benefício no que se refere a identificar problemas de usabilidade.

A.2 Metodologia

Nessa seção são apresentados os passos que foram realizados para planejar e conduzir a avaliação. Um aspecto essencial em toda avaliação de sistemas interativos é que ela seja adequadamente planejada. Caso contrário, corre-se o risco de consumir recursos sem obter os resultados esperados. Portanto, um bom planejamento aumenta as chances de sucesso da avaliação.

O framework DECIDE (SHARP; ROGERS; PREECE, 2007) foi utilizado para auxiliar no planejamento da avaliação. Ele define um conjunto de passos que contribuem para obtenção de resultados satisfatórios, conforme apresentado a seguir.

1. **D**eterminar as metas e objetivos da avaliação. Nessa avaliação, temos como meta avaliar a usabilidade oferecida pelo DengueME, visando a identificação dos possíveis problemas que o usuário possa encontrar e a recomendação de como solucioná-los. Portanto, é essencial verificar se a interface do DengueME apoia o usuário na criação e execução de modelos, e se apoia o modelador na criação de interfaces gráficas para novos modelos.

2. **E**xplorar questões específicas que se espera responder com a avaliação. A partir do objetivo inicial, foram definidas as seguintes questões específicas: 1) A interface gráfica ajuda o modelador a criar e parametrizar modelos? 2) A interface favorece o aprendizado de uso e memorização? 3) O software oferece eficiência de uso para criação e execução de modelos? 4) A interface oferece segurança de uso, prevenindo os usuários de cometer erros e auxiliando na recuperação quando eles ocorrem?

3. Es**C**olher os métodos e as técnicas que ajudarão a responder às questões específicas. Devido às limitações de recursos e indisponibilidade de usuários reais para participar da avaliação, optou-se por utilizar o método de avaliação heurística. Esse método foi escolhido por ter custo relativamente baixo, ser rápido e ter como foco encontrar problemas que prejudiquem a usabilidade do software (BARBOSA; SILVA, 2010a).

4. **I**dentificar questões práticas que precisam ser abordadas na avaliação. A seguir são apresentadas algumas questões práticas com as quais tivemos que lidar para conduzir essa avaliação.

    - Definir a quantidade de avaliadores: embora não seja o ideal, por restrições de recursos a avaliação foi conduzida por um único avaliador - o próprio autor desse documento.

    - Definir local da avaliação: a avaliação foi realizada no Laboratório de Engenharia e Desenvolvimento de Sistemas (LEDS), localizado no Instituto de Ciências Exatas e Aplicadas da Universidade Federal de Ouro Preto.

- Definir partes do software que serão avaliadas: um conjunto de tarefas foi definido visando analisar cenários de uso sob o ponto de vista dos dois principais cenários de uso do DengueME - usar um modelo existente na biblioteca para realizar simulações e criar um novo modelo com interface gráfica para sua utilização. Dessa forma, as principais partes do software que foram avaliadas são a janela principal do software, que inclui o editor de modelos, e o módulo *Model Builder*, que é usado para criação de interfaces para novos modelos.

- Definir classes de severidade: foram adotadas as mesmas classes de severidade apresentadas por Simone 2010a, a saber: (a) Problema Cosmético, nada muito sério, não precisa ser consertado caso não exista tempo no cronograma; (b) Problema Pequeno, o conserto deste problema tem baixa prioridade; (c) Problema Grande, importante de ser consertado e tem alta prioridade; (d) Problema Catastrófico, extremamente importante de ser consertado, pois se mantido, pode impedir o usuário de realizar as tarefas propostas pelo sistema.

5. **D**ecidir como lidar com questões éticas. Tendo em vista que somente foi utilizado o método de avaliação heurística, que não envolve usuários, não foi necessário lidar com aspectos relacionado à questões éticas.

6. Avaliar (**E**valuate), interpretar e apresentar os dados. Nessa etapa a avaliação é conduzida, os dados coletados são analisados e interpretados e os resultados apresentados. Tais informações serão detalhadas na seção seguinte, Resultados, que apresenta o relatório da avaliação heurística.

A.3 Resultados

Os resultados da avaliação heurística são apresentados a seguir, no formato de um relatório de avaliação, conforme modelo proposto por Simone 2010a.

A.3.1 Objetivos

A avaliação tem como meta analisar a usabilidade oferecida pela interface do DengueME, visando identificar possíveis problemas que o usuário possa encontrar e recomendar como solucioná-los. Fazem parte das questões que se espera responder com a avaliação: 1) A interface gráfica ajuda o modelador a criar e parametrizar modelos? 2) A interface favorece o aprendizado de uso e memorização? 3) O software oferece eficiência de uso para criação e execução de modelos? 4) A interface oferece segurança de uso, prevenindo os usuários de cometer erros e auxiliando na recuperação quando eles ocorrem?

A.3.2 Escopo da avaliação

Faz parte do escopo da avaliação analisar a usabilidade do DengueME, o que inclui fatores como facilidade de aprendizagem de uso e de memorização, eficiência de uso, e segurança de uso. O foco da avaliação foram os dois principais cenários de uso da ferramenta: realizar simulações a partir de modelos existentes na biblioteca e criar novos modelos com respectivas interfaces gráficas de parametrização. Desta forma, a avaliação envolveu principalmente as interfaces da janela principal, que inclui o editor de modelos e *wizards* para criação de modelos, e o módulo *Model Builder*, usado para criar interfaces para novos modelos.

A.3.3 Breve descrição do método

A avaliação heurística consiste na análise de partes da interface previamente escolhidas. O avaliador analisa cada uma das partes se baseando em um conjunto de heurísticas predefinido. O avaliador deve tentar encontrar pontos da interface que violem as heurísticas. Ao encontrar um problema, um grau de severidade e possíveis soluções devem ser atribuídos ao problema. Ao esgotar todas as heurísticas que foram definidas, o(s) avaliador(es) deve(m) consolidar os resultados e elaborar o relatório da avaliação.

A.3.4 Número e perfil dos avaliadores

A avaliação foi realizada por um único avaliador, Lucas Saraiva Ferreira, autor desse relatório e atualmente estudante de graduação em Sistemas de Informação pela Universidade Federal de Ouro Preto. O avaliador possui experiência em desenvolvimento de software e na ferramenta DengueME, sendo um dos seus desenvolvedores. Idealmente a avaliação deveria ser feita por pessoas diferentes daquelas que estão desenvolvendo o sistema, mas por restrições de recursos, isso não foi possível. Esse problema foi minimizado pois a avaliação foi realizada em um estágio inicial do projeto, quando o avaliador ainda não estava efetivamente envolvido com tarefas de desenvolvimento de software.

A.3.5 Heurísticas de usabilidade e problemas encontrados

Nessa seção são apresentadas as heurísticas utilizadas, e respectivos problemas encontrados, relacionados a cada heurística. Para cada problema encontrado são também apresentadas as seguintes informações: local, grau de severidade, defeito, solução.

1. **Visibilidade do status do sistema**. O sistema deve manter o usuário informado sobre o que está acontecendo - qual o atual estado que o sistema se encontra - através de feedback apropriado e em um tempo aceitável.

- **Problema 1**: A interface não informa o usuário se modificações feitas modelo estão ou não salvas. Diversas aplicações, como por exemplo o editor *Sublime Text*, oferecem um feedback visual como um '*' ao lado do nome do arquivo para indicar que existem alterações no arquivo aberto que ainda não foram salvas. **Local:** interface da janela principal e do *Model Builder*. **Severidade:** Problema Cosmético. **Recomendação:** Implementar a mesma estratégia que editores de texto usam. Toda vez que houver alguma mudança nos parâmetros do modelo, um asterisco deve aparecer ao lado do nome do modelo.
- **Problema 2**: Ao executar a operação *"RUN"*, para executar um modelo, nenhum feedback é fornecido para o usuário - nenhum texto ou janela abre para dizer qual o atual estado do programa. É impossível saber se o programa está carregando o modelo ou se algum erro ocorreu. **Local:** interface da janela principal. **Severidade:** Problema Grande. **Recomendação:** fornecer feedback por meio de mensagens a serem exibidas após a operação "Run"ter sido invocada. Por exemplo: Ao apertar o botão "Run", uma mensagem "Loading"poderia indicar que o modelo está sendo carregado, ou uma mensagem de erro deveria ser exibida se algo inesperado ocorreu.
- **Problema 3**: Ao utilizar a operação "Rename", o caminho absoluto do modelo mostrado no topo da ferramenta não é atualizado, gerando uma inconsistência. O usuário pode pensar que o nome não foi alterado corretamente e tentar alterar novamente, causando a repetição de uma ação. **Local:** interface da janela principal. **Severidade:** Problema Grande. **Recomendação:** atualizar o caminho absoluto na interface em tempo real. Caso o nome do projeto ou do modelo seja alterado, o caminho exibido na interface deveria ser atualizado automaticamente. A operação "Rename"está funcionando corretamente, o problema está ocorrendo devido à falta de feedback correspondente na janela da interface principal.

2. **Falar a linguagem do usuário**. A terminologia deve ser baseada na linguagem do usuário e não orientada ao sistema. As informações devem ser organizadas conforme o modelo mental do usuário.

   - **Problema 1**: O botão "Restore"não tem um rótulo claro o suficiente. O botão tem como função restaurar os parâmetros de um modelo para seus valores originais. **Local:** interface do editor de modelos. **Severidade:** Problema Pequeno. **Recomendação:** mudar o rótulo do botão para "Reset Model"ou "Default Values".
   - **Problema 2**: o termo "Run"não se aplica a linguagem dos futuros usuários do DengueME. O botão tem como funcionalidade executar um modelo baseado nos parâmetros, fazendo uso da plataforma TerraME. **Local:** menu principal

e barra de ferramentas. **Severidade:** Problema Pequeno. **Recomendação:** mudar o rótulo do botão para "Simulate Model"ou "Simulate".

- **Problema 3**: O termo "Clear"não parece ser claro o suficiente para o usuário entender seu funcionamento. O botão tem como função limpar das mensagens mostradas na *view Console*, que exibe a saída produzida pelo interpretador durante a execução de um modelo. **Local:** *view Console*. **Severidade:** Problema Pequeno. **Recomendação:** mudar o rótulo do botão para "Clear Messages"ou "Clear Terminal".

3. **Saídas claramente demarcadas**. O usuário controla o sistema, ele pode, a qualquer momento, abortar uma tarefa, ou desfazer uma operação e retornar ao estado anterior.

   - **Problema 1**: o usuário não consegue mover modelos entre projetos. Caso o usuário insira um novo modelo no projeto errado, ele tem que apagar o modelo e criar ele novamente ou ir até a pasta do *workspace* e mudar manualmente o modelo de lugar. **Local:** *view Project Explorer* na janela principal. **Severidade:** Problema Grande. **Recomendação:** permitir que o usuário consiga mover os modelos entre projetos utilizando a própria interface do DengueME.

   - **Problema 2**: caso o usuário cometa algum erro, não é possível desfazer a ação. Para corrigir qualquer erro cometido, o usuário precisa manualmente desfazer as ações anteriores que foram realizadas. **Local:** editor de modelos, *Model Builder*. **Severidade:** Problema Grande. **Recomendação:** disponibilizar recursos para recuperação de erros, como por exemplo desfazer as últimas ações realizadas na interface.

   - **Problema 3**: Uma vez na interface do *Model Builder*, ao se iniciar a criação de uma interface para um novo modelo, não é possível voltar a interface inicial do *Model Builder*. **Local:** *view Console*. **Severidade:** Problema Grande. **Recomendação:** permitir ao usuário retornar à interface principal (janela anterior) do *Model Builder*.

4. **Consistência**. Um mesmo comando ou ação deve ter sempre o mesmo efeito. A mesma operação deve ser apresentada na mesma localização e deve ser formatada/apresentada da mesma maneira para facilitar o reconhecimento.

   - **Problema 1**: A opção "About"no menu "Help"tem o mesmo ícone do botão "Help", localizado no canto direito do editor de modelos. Os dois realizam funções totalmente diferentes: o "About"apresenta informações sobre o software, enquanto o botão "Help"exibe informações sobre o modelo em edição. **Local:** menu da janela principal e editor de modelos. **Severidade:** Problema Pequeno. **Recomendação:** alterar o ícone do item de menu "About".

- **Problema 2**: as opções "Exit"e "Close Model"possuem o ícone, embora realizem operações distintas: enquanto a primeira fecha completamente a aplicação DengueME e a segunda fecha somente o modelo aberto naquele momento. **Local:** interface da janela principal. **Severidade:** Problema Grande. **Recomendação:** alterar o ícone da opção "Exit".

- **Problema 3**: o botão "Remove Group"na aba "Simulation"da interface do *Model Builder* não faz nada - ao ser clicado nenhuma ação é realizada. Por sua vez, na aba “Parameters”, o botão se comporta conforme esperado - remove um grupo de parâmetros. **Local:** interface do *Model Builder*. **Severidade:** Problema Pequeno. **Recomendação:** implementar a funcionalidade para remover um grupo de parâmetros e associá-la ao botão.

- **Problema 4**: o botão "Remove Group"na aba "Results"da interface do *Model Builder* não faz nada - ao ser clicado nenhuma ação é realizada. Por sua vez, na aba “Parameters”, o botão se comporta conforme esperado - remove um grupo de parâmetros. **Local:** interface do *Model Builder*. **Severidade:** Problema Pequeno. **Recomendação:** implementar a funcionalidade para remover um grupo de parâmetros e associá-la ao botão.

- **Problema 5**: na interface do *Model Builder*, ao se adicionar um novo parâmetro na interface de um modelo, é exibido um pequeno botão no canto direito que permite alterar o tipo do parâmetro. Quando o tipo do parâmetro é CSV, o botão não exibe as opções para alterar o tipo do parâmetro, um exemplo de inconsistência interna. **Local:** aba "Parameters"da interface do *Model Builder*. **Severidade:** Problema Grande. **Recomendação:** implementar a funcionalidade que permite alterar o tipo do parâmetro para todos os tipos possíveis.

5. **Prevenção de erros**. "Melhor que uma boa mensagem de erro é um design cuidadoso que possa prevenir esses erros"(NIELSEN; MOLICH, 1990). Por exemplo, ações definitivas, como remoção de objetos podem vir acompanhadas de mensagens de confirmação.

   - **Problema 1**: É possível criar uma interface para um modelo sem colocar qualquer informação. Caso o usuário faça isso, quando ele tentar parametrizar seu modelo no editor da interface principal, a ferramenta será interrompida (*crash*) por funcionamento não esperado. **Local:** editor da janela principal e *Model Builder*. **Severidade:** Problema Catastrófico. **Recomendação:** definir um conjunto de regras de validação para o *Model Builder*. Caso o usuário tente salvar um novo modelo sem definir pelo menos o conjunto mínimo de informações obrigatórias, a ação deve ser impedida e uma mensagem de erro deve ser exibida.

Além disso, o software deve ser corrigido para evitar o comportamento inesperado de quebra na aplicação.

6. **Reconhecimento ao invés de lembrança**. Evite acionar a memória do usuário o tempo inteiro, fazendo com que cada ação precise ser revista mentalmente antes de ser executada. Permita que a interface ofereça ajuda contextual, e informações capazes de orientar as ações do usuário – ou seja – que o sistema dialogue com o usuário.

   - **Problema 1**: o *Model Builder* é uma das principais funcionalidades do DengueME, e ela se encontra "escondida"em um menu. **Local:** interface da janela principal. **Severidade:** Problema Cosmético. **Recomendação:** incluir na barra de ferramentas um ícone de atalho para o *Model Builder*.
   - **Problema 2**: o *Model Builder* é uma ferramenta criada para ajudar modeladores a criar interfaces para seus modelos. O problema é que, ao entrar na ferramenta, não existe nenhum tipo de ajuda ou orientação para ensinar o usuário qual a finalidade e como usar a ferramenta. **Local:** interface do *Model Builder*. **Severidade:** Problema Grande. **Recomendação:** disponibilizar um sistema de ajuda dentro do *Model Builder* que tenha informações básicas sobre como usar a ferramenta, oferecer um tutorial que oriente o usuário sobre o que deve fazer ao entrar na ferramenta pela primeira vez.

7. **Flexibilidade e eficiência de uso**. O sistema precisa ser fácil para usuários leigos, mas flexível o bastante para se tornar ágil à usuários avançados. Essa flexibilidade pode ser conseguida com a permissão de teclas de atalhos, por exemplo.

   - Não foram encontrados potenciais problemas relacionados a essa heurística. A interface do DengueME oferece um conjunto de atalhos que permitem o acesso rápido às principais funcionalidades. Além disso, a interface do software possui poucas janelas e é bem simples de ser utilizada.

8. **Estética e design minimalista**. Evite que os textos e o design falem mais do que o usuário necessita saber. Os “diálogos” do sistema precisam ser simples, diretos e naturais, presentes nos momentos em que são necessários.

   - **Problema 1**: na janela "General Options"existe uma conjunto de interrogações ("?") perto dos campos que não fazem absolutamente nada. **Local:** interface da janela "General Options", que pode ser acessada a partir do principal, opção "Tools". **Severidade:** Problema Cosmético. **Recomendação:** remover as interrogações.
   - **Problema 2**: existe um botão chamado "Clone"na interface do Model Builder. Atualmente o botão não realiza nenhuma ação. Além disso, no meu entendimento,

cada modelo é totalmente diferente do outro, logo não existe necessidade desta funcionalidade. Uma cópia de um mesmo modelo não deveria existir. **Local:** interface do *Model Builder*. **Severidade:** Problema Pequeno. **Recomendação:** remover o botão "Clone".

9. **Ajude os usuários a reconhecer, diagnosticar e sanar erros**. As mensagens de erro do sistema devem possuir uma redação simples e clara que ao invés de intimidar o usuário com o erro, indique uma saída construtiva ou possível solução.

   - **Problema 1**: não existe mensagens de erro explicitas no sistema. Quando um modelo falha sua execução, a ferramenta TerraME mostra um erro no console, mas o mesmo não é nada explicito para quem está usando a ferramenta. **Local:** *view Console*. **Severidade:** Problema Grande. **Recomendação:** filtrar as mensagens de erro do TerraME e oferecer elas de forma mais amigável.

10. **Ajuda e documentação**. Um bom design deveria evitar ao máximo à necessidade de ajuda na utilização do sistema. Ainda assim, um bom conjunto de documentação e ajuda deve ser utilizado para orientar o usuário em caso de dúvida. Deve ser visível, facilmente acessada, e com oferecer uma ferramenta de busca na ajuda.

    - **Problema 1**: o *Model Builder* não tem nenhum sistema de ajuda. **Local:** *Model Builder*. **Severidade:** Problema Grande. **Recomendação:** criar um sistema de ajuda para o *Model Builder*.
    - **Problema 2**: a ferramenta não oferece nenhum tipo de documentação ou manual de uso. **Local:** interface da janela principal, *Model Builder*. **Severidade:** Problema Grande. **Recomendação:** fornecer um manual de uso e documentação de usuário.

A.4 Analise dos Resultados

A partir da avaliação foi possível perceber que a interface do DengueME de certa forma ajuda o modelador a criar e parametrizar modelos, mas ainda necessita de melhorias para melhor apoiar o usuário em diversos casos. Por exemplo, indicar se um modelo em edição está salvo, se um modelo está atualmente sendo executado, oferecer um sistema de ajuda. Em relação ao *Model Builder*, embora a interface seja a princípio simples e amigável, ela carece de um sistema de ajuda e documentação de apoio, como tutoriais e manuais de uso. Há várias necessidades de mudança na interface, que contribuiriam para melhorar a sua usabilidade. Mas um problema geral e mais grave é a completa ausência de um sistema de ajuda e documentação de uso.

Uma das questões levantadas se refere à eficiência de uso da interface. Embora existam atalhos, que contribuem para melhorar a eficiência de uso, não foi possível responder

essa pergunta de forma efetiva, somente por meio dessa avaliação. Seria interessante a participação de usuários, com diferentes perfis e níveis de experiência, para melhor avaliar esse aspecto de usabilidade. O mesmo vale para aprendizagem de uso e memorização, embora a disponibilidade de um sistema de ajuda possa contribuir para ambos os fatores.

Em relação à segurança de uso, a interface apoia o usuário em alguns aspectos, mas falha de forma catastrófica em outros. Por exemplo, a interface impede o usuário de inserir caracteres inválidos em nomes de arquivos e pastas. Mas ela permite ao modelador criar um novo modelo cuja interface é vazia, o que leva a uma falha grave no software (*crash*) quando um usuário tenta abrir esse modelo vazio.

A.4.1 Limitações da Avaliação

A avaliação heurística é um método eficiente que permite obter resultados satisfatórios a um custo relativamente baixo. Mas é importante destacar que nem todos os aspectos no que se refere à qualidade de uso de um sistema podem ser capturados utilizando somente métodos analíticos como a avaliação heurística. Para uma avaliação mais abrangente é fundamental envolver os usuários reais da aplicação.

Uma das limitações dessa avaliação é o fato de ela ter sido realizada por somente um avaliador, quando o recomendado é que ela seja feita por aproximadamente 5 avaliadores. A participação de um número maior de avaliadores contribui para identificar um número maior de problemas de usabilidade e também para recomendar soluções de forma mais adequada. Outro aspecto relevante que afeta o resultado é a escolha das heurísticas. Em avaliações futuras, heurísticas específicas voltadas para a modelagem e simulação podem ser definidas e utilizadas.

A.5 Considerações finais

A usabilidade do software DengueME foi avaliada através do método de avaliação heurística. A interface do software foi inspecionada utilizando um conjunto de dez heurísticas, com o intuito de responder a um conjunto específico de perguntas relacionadas à qualidade de uso oferecida pela interface do sistema, em particular, usabilidade. Foi possível responder parte das perguntas levantadas e identificar vários problemas de usabilidade somente com o método de avaliação heurística. Para uma avaliação mais abrangente, recomenda-se envolver usuários e utilizar métodos de observação de uso.

A avaliação permitiu identificar diversos problemas de usabilidade que os usuários poderiam vir a encontrar ao interagir com a interface do DengueME. O método de avaliação heurística se mostrou apropriado, tendo em vista que o objetivo da avaliação era encontrar problemas de usabilidade na versão *beta* do DengueME com o intuito de desenvolver melhorias de interface. Os problemas encontrados e as recomendações foram utilizados

para guiar o desenvolvimento de novas versões do software, incluindo a versão 1.0 do DengueME.

Os métodos de avaliação da qualidade de uso e de experiência de uso devem ser vistos como algo complementar e necessário ao desenvolvimento de sistemas interativos. A avaliação heurística, sozinha, não é suficiente para garantir uma boa qualidade de uso para os sistemas. Mas ela é eficiente ao permitir identificar vários problemas potenciais. Por fim, as melhorias de qualidade obtidas a partir de tais resultados podem ser analisadas com a aplicação de métodos de avaliação que envolvam os usuários.