

**UM FRAMEWORK CONCEITUAL DE
MODELAGEM E SIMULAÇÃO: ESTUDO DE
CASO NA REENGENHARIA DA FERRAMENTA
DENGUEME**

LUCAS SARAIVA FERREIRA

UM FRAMEWORK CONCEITUAL DE
MODELAGEM E SIMULAÇÃO: ESTUDO DE
CASO NA REENGENHARIA DA FERRAMENTA
DENGUEME

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

ORIENTADOR: CLODOVEU AUGUSTO DAVIS JUNIOR

Belo Horizonte
Setembro de 2020

Ferreira, Lucas Saraiva.

F383f

Um framework conceitual de modelagem e simulação
[manuscrito]: estudo de caso na reengenharia da ferramenta
Dengueme / Lucas Saraiva Ferreira – 2020.
xvi, 82 f. il.

Orientador: Clodoveu Augusto Davis Junior.
Dissertação (mestrado) – Universidade Federal de Minas
Gerais, Instituto de Ciências Exatas, Departamento de Ciência
da Computação.
Referências: f.77-82.

1. Computação – Teses. 2. Modelagem de dados – Aspectos
ambientais – Teses. 3. Sistemas de informação geográfica –
Teses. 4. Dengue – Teses. I. Davis Junior, Clodoveu Augusto.
II. Universidade Federal de Minas Gerais; Instituto de Ciências
Exatas, Departamento de Ciência da Computação. III. Título.

CDU 519.6*72(043)



UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Um framework conceitual de modelagem e simulação: estudo de caso na reengenharia da ferramenta DengueME

LUCAS SARAIVA FERREIRA

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. CLODOVEU AUGUSTO DAVIS JÚNIOR - Orientador
Departamento de Ciência da Computação - UFMG

PROF. ANDRÉ CAVALCANTE HORA
Departamento de Ciência da Computação - UFMG

PROFA. RAQUEL MARTINS LANA
FIOCRUZ - RJ

PROFA. ANA PAULA COUTO DA SILVA
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 25 de Setembro de 2020.

Resumo

Arboviroses como a dengue, zika, chikungunya entre outras são um problema de saúde pública do Brasil e do mundo, sendo a Dengue, por exemplo, endêmica em todos continentes com exceção da Antártida. O crescimento da popularidade da computação e o avanço das tecnologias existentes faz com que diversas áreas de estudo sejam beneficiadas de diferentes formas. A modelagem e simulação computacional é uma ferramenta da área de computação que consegue impactar de forma positiva problemas de decisão, prevenção e predição envolvendo doenças. Neste contexto, este trabalho tem como objetivo a apresentação de um *framework* para criação de ferramentas de modelagem e simulação de arboviroses. O *framework* foi criado com base em ferramenta já existente e nas necessidades encontradas na literatura. Foi realizada uma pesquisa e análise de ferramentas de modelagem já existentes, listando pontos fortes e fracos com intuito de se chegar em um *framework* completo. O mesmo pode ser usado para criação de ferramentas do zero ou refatoração de já existentes, como é o caso do objeto de exemplo deste trabalho, DengueME, que teve toda sua arquitetura modificada para se adequar ao *framework* produzido no trabalho. Estudos de caso são apresentados envolvendo exemplos de diferentes tipos de usuário, demonstrando o potencial de ferramentas baseadas no *framework*. Com o trabalho conseguimos demonstrar que é possível implementar o *framework* em ferramentas e demonstramos as diversas funcionalidades auxiliares que ele traz.

Palavras-chave: Dengue, Modelagem, Framework, DengueME.

Abstract

Arboviral diseases such as dengue, zika, chikungunya and others constitute a public health problem in Brazil and abroad, as dengue, for instance, is endemic in every continent except Antarctica. The increased popularity of computing and the advancement of related technologies brings benefits to several fields of study, in various aspects. Computational modeling and simulation is a computer science tool that can provide positive impact on decision problems, for preventing and predicting disease spread. In this context, this work aims to present a framework for the creation of modeling and simulation tools directed at arboviral diseases. The framework is based on existing tools and considers user needs presented in the literature. A survey of existing modeling tools was conducted, listing their strengths and weaknesses, to reach a broad set of features. The framework can be used to create tools from scratch or to refactor existing ones, such as in DengueME, a tool whose architecture was modified to fit the proposed framework. Case studies are presented, involving examples for different types of users, to show the framework's potential for developing new tools. This work demonstrated that it is possible to implement such tools, and shows various complementary functions enabled by it.

Keywords: Dengue, Modeling, Framework, DengueME.

Lista de Figuras

2.1	Interface do SEARUMS++. Fonte: [Rao, 2016]	10
2.2	Interface do NetLOGO. Fonte: [Jacobson et al., 2013]	12
2.3	Fluxo do AGWA. Fonte: [Burns et al., 2004]	15
3.1	Framework	26
3.2	Associação processo de modelagem de Brooks e o <i>Framework</i>	27
3.3	Pacote de modelo.	38
3.4	Caso de uso.	39
3.5	Caso de uso.	41
4.1	Arquitetura DengueME original.	46
4.2	Estado atual do DengueME.	52
5.1	Configurações DengueME.	56
5.2	Tela inicial construtor de pacote de modelos DengueME.	57
5.3	Informações básicas do modelo.	57
5.4	Criação da interface de parametrização.	58
5.5	Interface de parametrização pronta.	58
5.6	Interface de saída pronta.	59
5.7	Interface de saída pronta.	59
5.8	Prototipo construtor do editor de dados.	60
5.9	Biblioteca de modelos.	61
5.10	<i>Dashboard</i> com interface de parametrização do modelo.	61
5.11	Mockup editor de dados.	62
5.12	Mockup módulo de simulação.	63
5.13	Resultados do modelo.	64
5.14	Mockup analisador de resultados.	64
5.15	Mockup calibrador de modelo.	65
5.16	Editor de modelo.	66

Lista de Tabelas

2.1	<i>Features</i> por ferramenta	19
-----	--	----

Sumário

Resumo	vii
Abstract	ix
Lista de Figuras	xi
Lista de Tabelas	xiii
1 Introdução	1
1.1 Motivação	5
1.2 Objetivos e perguntas de pesquisa	5
1.3 Contribuições	6
1.4 Organização do Trabalho	7
2 Trabalhos Relacionados	9
2.1 SEARUMS++	9
2.2 P2PCoDiGeosim	10
2.3 MalariaTools	11
2.4 NETLOGO	11
2.5 DengueME	13
2.6 AGWA	13
2.7 Modelo Armadilha de Mosquito	14
2.8 Comparativo	16
2.9 Considerações Adicionais	17
3 Metodologia	21
3.1 Definições e Requisitos	21
3.2 Projeto do Framework	26
3.2.1 Módulos do Framework	29

3.2.2	Outros componentes do framework	36
3.3	Fluxo de uso	37
3.3.1	Modelador programador	38
3.3.2	Modelador não programador	40
3.4	Requisitos funcionais	42
4	Reengenharia e Extensão do DengueME	45
4.1	Reengenharia	47
4.2	Modularização	48
4.3	Extensões e Implementações Pendentes	49
4.4	Documentação	51
4.5	Considerações	52
5	Estudos de Caso	55
5.1	Caso de Uso 1	55
5.2	Caso de Uso 2	66
5.3	Comparativo	69
6	Conclusão e trabalhos futuros	71
6.1	Ameaças à Validade	74
	Referências Bibliográficas	77

Capítulo 1

Introdução

Arboviroses são doenças causadas por vírus transmitidos por vetores artrópodes. A transmissão dos arbovírus ocorre por meio da picada, como por exemplo durante a realização da alimentação por sangue das fêmeas do mosquito *Aedes* [Kuno & Chang, 2005]. O cenário atual indica que todos os continentes, com exceção da Antártida, são endêmicos a algum arbovírus [Rust, 2012; Gould et al., 2017a].

As arboviroses são doenças negligenciadas e não eram vistas como ameaças em escala global por várias décadas [Wilder-Smith et al., 2017a]. Nas últimas 5 décadas essa noção começou a se inverter e as arboviroses passam a ser percebidas como um problema para a sociedade mundial [Patterson et al., 2016]. O aumento da densidade populacional, a adaptação dos vetores em ambientes urbanos e as mudanças climáticas são alguns dos fatores que contribuem para a reemergência em massa de arboviroses e, portanto, o seu destaque no mundo [Gould et al., 2017b; Gubler, 2002; Kilpatrick & Randolph, 2012; Wilder-Smith et al., 2017a]. Outro fator crucial para a disseminação das arboviroses é a facilitação da mobilidade humana [Gould et al., 2017b]. Historicamente, o *Aedes* chegou ao continente americano via transporte marinho [Smith et al., 1984]. A dengue é um exemplo clássico de doença cuja disseminação é intensificada pelo movimento de pessoas infectadas, assim como o transporte passivo de mosquitos. A facilitação de deslocamento provinda do avanço tecnológico e urbano acaba por causar impacto em como somos afetados por diversas doenças [Smith et al., 2014].

No Brasil, as principais arboviroses em circulação são a dengue, chikungunya e zika [Donalisio et al., 2017]. No ano de 2019, foram notificados ao Ministério da Saúde 1.439.471 casos de dengue, 110.627 de chikungunya e 9.813 de zika, segundo dados do boletim epidemiológico número 22, de 2019 [Ministério da Saúde, 2017], um número absurdamente maior que o número de casos em 2017 [Ministério da Saúde, 2017]. Podemos ver aqui um dos efeitos da negligência anteriormente citada. Esforços

no controle e prevenção das arboviroses têm sido realizados, mas ainda sem grande êxito [Wilder-Smith et al., 2017a], uma vez que a dinâmica de transmissão dessas doenças é de extrema complexidade.

É notório que dentre os diversos fatores para a transmissão e propagação de arboviroses estão fatores espaço-temporais, associados às características do lugar e à dinâmica de propagação [Higa, 2011; Horta et al., 2014; Farinelli et al., 2018; Ng et al., 2017; Karl et al., 2014]. Esses fatores nem sempre são facilmente associados ou discerníveis em cada caso em particular. Para se descobrir possíveis fatores que influenciam a transmissão é necessário coletar e analisar dados baseados nos diversos sistemas que emergem dos estudos dessas e de diversas outras doenças. Por exemplo, Horta et al. [2014] demonstram a correlação entre dengue e crescimento urbano.

Em geral, é difícil compreender sistemas complexos apenas pelo estudo de suas entidades, pois um sistema complexo é composto por partes que a interação entre elas leva a propriedades e comportamentos emergentes [Baianu, 2011]. Um sistema complexo possui diversas partes que interagem entre si e que podem influenciar nos resultados de diferentes formas, sendo difícil entendê-lo por inteiro [Baianu, 2011]. Por isso, para auxiliar no estudo desses fenômenos, podemos fazer uso de ferramentas como modelos matemáticos, simulação computacional e Sistemas de Informação Geográficos (SIG). Um modelo pode ser visto como uma representação simplificada da realidade (sistema em estudo). A simulação consiste na execução do modelo imitando um fenômeno real em determinado intervalo de tempo [Sokolowski & Banks, 2010]. SIGs são ferramentas que coletam, armazenam, processam, transformam e exibem dados espaciais ou não espaciais [Coppock & Rhind, 1991]. A integração de dados espaciais a modelos é uma ideia antiga, como verificado em Ostfeld et al. [2005], em que já se fala da importância de dados e observações espaciais para criação de simulações próximas da realidade. Além disso, é desejável estudar diversos aspectos de um mesmo sistema complexo, não se limitando as análises a somente um fator específico.

A aplicação de SIG em saúde pública é de grande valor, pois traz melhorias e benefícios para a prestação de serviços [Carvalho et al., 2000; Fletcher-Lartey & Caprarelli, 2016]. Ao combinar análises espaciais, algoritmos sofisticados, modelagem e Geoestatística tem-se um instrumento forte para compreender os padrões de disseminação e auxiliar no combate e na prevenção de doenças [Higgs, 2004; Guo et al., 2005]. Fletcher-Lartey & Caprarelli [2016] apresentam diversos casos de sucesso para o atual cenário da aplicação de SIG na saúde e demonstram que, em suma, as ferramentas de SIG são utilizadas para monitoramento ou visualização de dados pré-processados.

A preparação de um modelo para simulação é composta de diversas etapas. Podemos enumerá-las como: (i) definir o sistema a ser estudado; (ii) definir quais as variáveis

envolvidas no modelo; (iii) implementar o modelo em linguagem de programação; (iv) parametrizar o modelo; (v) simular e avaliar os resultados; (vi) calibrar o modelo; (vii) repetir visando aprimoramento. As etapas (i) e (ii) fazem parte da competência do modelador, enquanto as etapas (iii), (iv), (v) e (iv) necessitam de conhecimentos em diferentes áreas, como programação, sistemas geográficos, etc.. Por exemplo, na etapa de implementar o modelo, se necessário, atrelar dados geográficos e não geográficos não é uma tarefa trivial e se mantém como um desafio mesmo com a grande quantidade de estudos na área. Especificamente, a incorporação eficiente de dados geográficos aos modelos ainda é uma lacuna na ciência [Kirby et al., 2017]. A obtenção de dados geográficos é apenas uma das etapas para sua aplicação em modelos epidemiológicos; em geral, transformações e processamento se fazem necessários. Essas operações podem ser tanto mais básicas e simples de serem realizadas, como também podem ser complexas e exigir certo aprendizado e domínio de técnicas específicas por parte dos modeladores. A complexidade desse tipo de análise pode ser observada no passo a passo metodológico descrito em Lana et al. [2018]. Consiste, portanto, em um esforço considerável e que precisa ser despendido devido à inexistência de ferramentas, como *frameworks* e softwares, que facilitem essa ou outras etapas do processo de modelagem e simulação. Neste sentido, o tempo que o modelador poderia estar dedicando à sua área de estudo (o modelo) é despendido na preparação, transformação e incorporação dos dados ao modelo [Riley et al., 2015]. Essas etapas podem também ser resumidas em: Modelagem, Simulação, Resultados e Conhecimento [Sokolowski & Banks, 2010].

Diante disso, é possível perceber que existe uma lacuna que torna o processo de uso de modelos computacionais uma tarefa complexa e demorada, especialmente se tratando de modelos envolvendo aspectos geográficos, que são de grande importância. Shaw [2012] demonstra que dados geográficos e sistemas geográficos têm grande impacto e importância para a saúde pública. Dessa forma, o desenvolvimento de ferramentas e/ou *frameworks* de software que auxiliem em diversos pontos do processo de modelagem e simulação, e que possam facilitar a incorporação de dados e aprimoramento de modelos epidemiológicos para arboviroses e diversos outros agravos se faz necessário [Ostfeld et al., 2005].

A criação de ferramentas, *frameworks* e uso da computação para modelagem e simulação tem sido um tema bastante abordado, dado seu impacto positivo nessa atividade [Carroll et al., 2014; Baker et al., 1995; Friede et al., 1995; Fletcher-Lartey & Caprarelli, 2016]. O uso de ferramentas traz diversos benefícios, que vão desde acelerar o processo de estudo dos modelos à criação de novos modelos [Friede et al., 1995]. Uma tendência é a criação de ferramentas restritas, que têm o objetivo de cobrir um estudo de caso específico ou somente um tipo de modelo. Essas ferramentas são úteis para os

estudiosos interessados naquele caso ou fenômeno, mas carecem em oferecer liberdade para o usuário explorar outros casos e expandir seus estudos sobre o fenômeno [Griffin et al., 2010; Zeilhofer et al., 2009]. Com base nisto, novo ferramental começou a ser pensado, como por exemplo o DengueME.

DengueME é um *framework* de software produzido em 2016 pelo laboratório LEDS - UFOP [de Lima et al., 2016]. A ferramenta se encontra disponível de forma gratuita no Github ¹, mas carece de diversas melhorias. Dentre essas melhorias está a definição e descrição de seu *framework*, de modo que outros possam utilizá-lo ou reproduzi-lo. Existe espaço para expansão de todos módulos já implementados e refinamento das ideias embutidas no sistema. Por exemplo, atualmente é possível realizar a modificação das regras dos modelos sem afetar outras partes do sistema graças à modularização da ferramenta. Outro exemplo é que a ferramenta atualmente suporta cerca de 10 tipos de entradas de dados. Esses tipos podem ser expandidos e melhor mapeados, para que, assim, seja possível cobrir uma gama maior de modelos. Todos estes pontos de expansão e melhoria podem ser identificados tendo a descrição de um *framework* e o acoplando a necessidades encontradas na literatura.

Por outro lado, existem ferramentas extremamente genéricas que atendem a diversos propósitos e podem ser utilizadas na modelagem e simulação. Um exemplo disto é a ferramenta AGWA ², um plugin de modelagem e simulação de *Watersheds* para o ArcGIS, criada para auxiliar no preparo dos dados, execução do modelo e apresentação dos resultados [Burns et al., 2004]. Este tipo de ferramenta pode levar a uma curva de aprendizado maior, pois, além de entender do uso do plugin, o usuário precisa entender como utilizar o básico da ferramenta, do contrário não conseguirá realizar as etapas básicas. A exemplo do AGWA, para se utilizar a ferramenta é necessário o carregamento e tratamento de diversos mapas e dados temporais, o que é uma tarefa simples para pessoas familiarizadas com SIG, mas para um modelador especialista em epidemiologia pode exigir um esforço que possivelmente poderia ser reduzido.

É muito importante conseguir balancear a flexibilidade de um *framework* e a dificuldade em seu uso. Ao mesmo tempo que se deseja entregar ao usuário algo com algum grau de generalidade, é preciso medir o quanto isso dificulta a tarefa de modelagem e simulação. Oferecer opções demais pode ser um problema tanto quanto oferecer opções de menos. Esse balanço é procurado neste trabalho e um *framework* é proposto nesse sentido. O *framework* é composto de 9 módulos que podem operar separadamente ou em conjunto. Cada módulo visa cobrir um campo que possa apoiar na modelagem e na simulação de modelos epidemiológicos. Nesse contexto, esta dissertação sugere

¹Github dengueME: <https://github.com/ufopleds/dengueME>

²AGWA: <https://www.tucson.ars.ag.gov/agwa/>

a definição formal, exploração, explicação e aplicação de um *framework* que consiga auxiliar no processo de modelagem e simulação. Além disso, a ferramenta dengueME foi modificada para atender ao *framework* aqui definido.

1.1 Motivação

Uma das principais motivações para esse trabalho foi a emergência e reemergência de diversas arboviroses como um problema de saúde pública em escala global, além da chance de outras arboviroses, atualmente não endêmicas, se tornarem uma ameaça maior [Gould et al., 2017b]. A avaliação e criação de novas ferramentas para auxiliar no estudo das arboviroses se faz importante para entender e estudar mais a fundo o complexo sistema em que estas doenças estão inseridas.

O aumento do poder computacional, associado à redução dos custos de hardware para processamento, além da maior possibilidade de acesso a estes recursos, contribuiu para a ampliação do uso de ferramentas para diversos estudos [Sokolowski & Banks, 2010]. Atualmente temos ferramentas de SIG *open source* poderosas, como o QGIS ³. A comunidade *open source* cresce rapidamente e a utilização desse tipo de ferramenta permite maior colaboração de terceiros [Weber & Weber, 2004]. Outras ferramentas para modelos específicos são apresentadas na revisão bibliográfica deste trabalho, as quais também representam uma grande ajuda a modeladores e estudiosos. A criação de um *framework* que permita que a comunidade crie novas ferramentas de modelagem e simulação que sejam flexíveis, que suportem dados geográficos e diversas plataformas de modelos é o principal diferencial deste trabalho.

O uso direto de ferramentas que apoiam a modelagem e simulação por parte dos especialistas em saúde pública, em geral, não é natural e intuitivo como necessário. Isso justifica o envolvimento da Computação para buscar uma melhor estruturação e definição dos aspectos do problema que dependem de modelagem, representação, dados, análises e apresentação de resultados visuais. Explorar formas de usar ferramentas e *frameworks* para apoiar atividades de planejamento, avaliação de cenários e tomada de decisão em saúde pública é outra motivação desta dissertação de mestrado.

1.2 Objetivos e perguntas de pesquisa

Esta dissertação tem como objetivo a produção de um *framework* que sirva de base para criação de *frameworks* de software de modelagem e simulação para diversos campos

³QGIS: https://www.qgis.org/pt_BR/site/

de estudo. O *framework* deve ser flexível quanto às suas entradas, saídas e modelos aceitos. Além disso, é necessário explicitar nele como dados geográficos seriam incluídos no mesmo. Para este trabalho focamos na modelagem de Arboviroses, uma vez que a ferramenta a ser usada de objeto para os casos de uso, o dengueME, tem este papel.

Os objetivos específicos do trabalho proposto incluem:

1. Estudo sobre ferramentas de modelagem epidemiológica que utilizem principalmente dados ou aspectos geográficos.
2. Identificação das lacunas existentes nos atuais sistemas de apoio a modelagem e simulação.
3. Definição formal de um *framework* para auxiliar na criação de ferramentas para modelagem e simulação, incluso modelagem e simulação de arboviroses.
4. Definição e execução de um estudo de caso com modelos para demonstração do funcionamento do *framework*.
5. Modificação da ferramenta dengueME para se adequar ao *framework* proposto.

Algumas questões de pesquisa foram levantadas para guiar este trabalho. Busca-se responder a essas questões ao longo do trabalho, de modo a cumprir os objetivos colocados.

Questão 1: As ferramentas de modelagem e simulação tem grande variação de funcionalidades e oferecem liberdade ao usuário?

Questão 2: Que etapas do processo de modelagem e simulação podem ser acopladas a um *framework*?

Questão 3: Que lacunas deixadas por outros softwares/*frameworks* podem ser cobertas?

Questão 4: Quais os impactos de uma ferramenta de modelagem e simulação que oferece maior flexibilidade ao usuário?

1.3 Contribuições

Durante o trabalho foi gerado um quadro comparativo entre ferramentas de modelagem e simulação usadas na revisão bibliográfica, elencando algumas funcionalidades e discutindo os pontos fortes e fracos de cada uma delas. Um estudo

de associação do processo de modelagem e simulação e uma ferramenta de modelagem e simulação foi realizado para demonstrar quais partes deste processo podem ser acoplados a uma ferramenta e como eles estão *representados* dentro dela. Uma reengenharia do DengueME foi feita para adequar o software ao framework (incompleto) para demonstrar as novas interações que ele ofereceria a diferentes tipos de usuário.

O trabalho conta com uma descrição completa do processo de reengenharia do DengueME e demonstração do resultado final aplicado a um caso de uso de modelagem e simulação usando um modelo real. Ao final, os pontos negativos e positivos desta mudança são elencados demonstrando as melhorias que a modificação na ferramenta trouxe.

1.4 Organização do Trabalho

Esta dissertação se divide da seguinte forma. O Capítulo 1 contém uma introdução ao tema, o objetivo e motivação do trabalho. O Capítulo 2 consiste na revisão bibliográfica, focada em outras ferramentas de modelagem e simulação para fins de comparação com o *framework* aqui produzido, e também apresenta as diferenças entre as ferramentas da literatura e o *framework* proposto. O Capítulo 3 apresenta a metodologia do trabalho, especificando o *framework* conceitual em si, seus módulos, o fluxo de uso para diferentes usuários e suas limitações. O Capítulo 4 apresenta as modificações realizadas sobre a ferramenta DengueME para se adequar ao *framework*. O Capítulo 5 consiste em um estudo de caso utilizando a ferramenta *DengueME*, a partir do *framework* proposto. Por fim, o Capítulo 5 traz a conclusão do trabalho e apresentação de trabalhos futuros, além das limitações encontradas no que foi produzido.

Capítulo 2

Trabalhos Relacionados

Este Capítulo apresenta algumas ferramentas de modelagem e simulação conhecidas e que serviram de base para a elaboração do *framework* conceitual deste trabalho.

2.1 SEARUMS++

SEARUMS (Studying Epidemiology of Avian Influenza Rapidly Using Modeling and Simulation) é uma ferramenta de código aberto para modelagem espacialmente explícita baseada em agentes voltada para o estudo da H5N1 criada em 2006 [Rao et al., 2006]. A ferramenta apoia as tarefas de modelagem, simulação e análises de resultados contando com uma interface gráfica e algumas opções de visualização da saída do modelo. Uma melhoria para o SEARUMS, o SEARUMS++, foi publicada em 2016 visando aumentar o desempenho da execução em paralela dos modelos [Rao, 2016]. A interface da ferramenta conta com diferentes visualizações e funcionalidades, figura 2.1.

Uma das principais inovações do SEARUMS é a possibilidade de uso de alguns dados geográficos, tais como latitude, longitude e fluxos migratórios. Para que o usuário possa usar os dados migratórios, é necessário que eles descrevam pontos de migração e contenham datas de chegada e saída do bando. Além disso, é preciso que um ciclo completo de migração seja descrito pelos dados [Rao, 2016]. Nem sempre os dados brutos contêm as informações necessárias para o funcionamento de modelo. Uma vez que a ferramenta não conta com uma documentação especializada internamente é difícil para o usuário ter a certeza de como parametrizar os dados.

Um dos principais limitantes da ferramenta é a falta de opções de modelos, sendo essa focada somente no modelo no qual foi programada para funcionar. Além disso, os dados de entradas são restritos e sem muitas explicações na ferramenta em si. Por fim, o fato de ela ser restrita a somente uma doença tira sua flexibilidade. A forma de

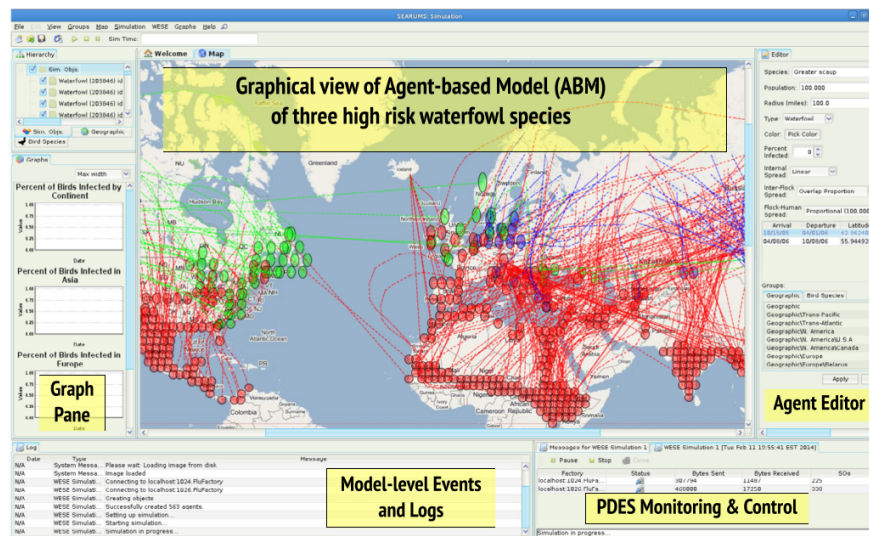


Figura 2.1. Interface do SEARUMS++. Fonte: [Rao, 2016]

resolver este problema seria a criação de um módulo separado que permita a inserção de diferentes modelos.

2.2 P2PCoDiGeosim

Person to Person Communicable Disease Geosimulation (P2PCoDiGeosim) Chen et al. [2014] é uma ferramenta de parametrização e simulação para doenças transmissíveis via contato. A ferramenta simula um modelo de 5 estágios: Suscetível, Exposto, Infectado, Recuperado e Morto. O usuário entra com diversos parâmetros no modelo epidêmico e dados espaciais, sendo esses dados da atividade populacional e mobilidade coletados na região metropolitana de Quebec, Canada.

A ferramenta consegue fazer a junção dos dados espaciais mencionados com um modelo matemático de transmissão, aumentando a complexidade do modelo como um todo, mas trazendo resultados mais consistentes. O autor afirma que essa integração (SIG + Modelo) é uma grande contribuição para a saúde pública, auxiliando na tomada de decisão sobre intervenção e prevenção [Haddad et al., 2016].

O P2PCoDiGeosim não oferece flexibilidade para o usuário quanto aos dados de entrada, pois o mesmo não permite que o usuário trate os dados espaciais através da ferramenta e possa usar a saída em qualquer modelo que desejar, toda a entrada e saída é restringindo a um modelo específico implantado na ferramenta. O usuário não tem liberdade para modificar o modelo matemático ou alterar suas saídas. A ferramenta não oferece nenhum tipo de customização de interface de parametrização e não conta

com uma biblioteca de modelos.

Sendo esta uma ferramenta que explora o uso de dados geográficos em um modelo, seria interessante oferecer mais opções ao usuário de forma que mais potencial seja obtido através da ferramenta.

2.3 MalariaTools

MalariaTools é uma ferramenta de parametrização para o modelo de malária criado por Griffien et al. em 2010 [Griffien et al., 2010]. A interface de parametrização foi criada posteriormente por Danail Stoyanov e lançada para download em 2012 [London, 2010]. A interface permite que o usuário utilize um conjunto predefinido de dados para simular diversos aspectos do vetor e da transmissão da malária. Alguns parâmetros podem ser modificados, outros são já determinados. Atualmente, a ferramenta simula somente um modelo que vem sendo melhorado ao longo dos anos [London, 2010]. O modelo embutido na ferramenta descreve a dinâmica de transmissão da doença representando as pessoas como indivíduos e os mosquitos como população ou nuvens de mosquitos. Ele leva em conta diversas variáveis (datas, taxas de transmissão, idade, porcentagem de vetores) e, inclusive, suporta definição de intervenções. A figura 2.2 mostra a interface de parametrização da ferramenta.

É possível criar diversos cenários do mesmo modelo dentro da ferramenta, variando somente os parâmetros. A ferramenta cumpre seu objetivo e tem resultados interessantes em relação à malária na África [Griffien et al., 2010]. Uma das principais limitações do *MalariaTools* é a dependência de somente um modelo, o qual não pode ser modificado diretamente pelo usuário e somente os criadores da ferramenta têm acesso para manutenções, calibrações ou mudanças. O usuário fica impedido de usar os resultados de suas simulações para melhorar o modelo matemático ou mudar aspectos gerais dele. O conjunto de saídas oferecidas pela ferramenta é bem restrito, os gráficos são predefinidos e os mapas funcionam somente para algumas cidades (consequência do modelo ser fechado a modificações de modeladores que tenham a ferramenta).

2.4 NETLOGO

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. Para se criar os modelos utiliza-se a linguagem *LOGO*, que permite a criação de modelos baseados em agentes. Através

dela é possível descrever o comportamento dos agentes envolvidos no modelo e suas variáveis de influência. 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). *NetLogo* conseguiu construir uma grande comunidade de modeladores interessados em modelagem por agentes ao permitir que os usuários tornem seus modelos públicos. A biblioteca de modelos é um dos módulos descrito nesse trabalho para incorporar ao *framework*.

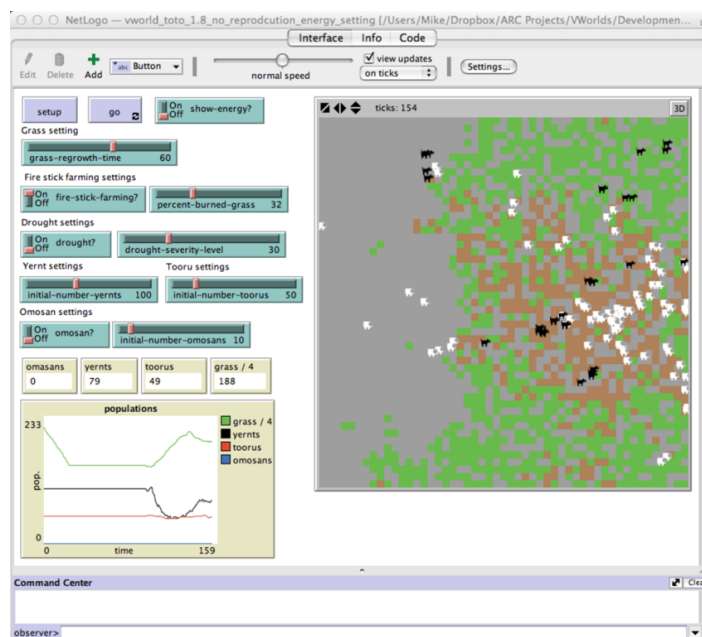


Figura 2.2. Interface do NetLOGO. Fonte: [Jacobson et al., 2013]

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 o construtor de interfaces 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, pois existe a curva de aprendizado do uso da interface.

Uma das limitações do *NetLogo* é a falta de flexibilidade quanto à linguagem usada na construção dos modelos, limitando o poder e variedade dos modelos ao paradigma de modelagem suportado pela linguagem LOGO. Existem linguagens como R e *TerraML* que permitem a confecção de modelos mais complexos, com uma variedade maior de paradigmas de modelagem e tipos de *inputs*. Oferecer uma opção para que modelos de diferentes linguagens/plataformas de simulação sejam executados pelo *framework* de software é um ponto necessário, mantendo assim a alta flexibilidade da ferramenta.

2.5 DengueME

DengueME (Dengue Modelling Environment) é uma ferramenta de modelagem e simulação da dengue disponível para Windows, Linux e Mac produzida pelo laboratório LEDS - UFOP [Ferreira, 2017]. A ferramenta foca em auxiliar o modelador a parametrizar e simular modelos de transmissão e ecologia do vetor da dengue. Atualmente se encontra na versão 1.0, oferecendo suportes a modelos escritos em TerraML e R, além disso, a ferramenta oferece ao usuário um construtor de interface, seguindo a mesma linha do NetLogo [de Lima et al., 2016].

O principal diferencial da ferramenta está em sua composição de módulos, aos quais permite fácil alteração e manutenção do código fonte da ferramenta. DengueME não tem um modelo fixo implementado, deixando livre ao modelador carregar qualquer modelo de Dengue que siga seus padrões de entrada e saída. Tendo o modelo, o usuário pode criar sua própria interface de parametrização e definir as visualizações para a saída.

É possível encontrar uma biblioteca de modelos na ferramenta, a biblioteca permite que os modelos sejam hospedados no Github junto de sua interface de parametrização. Atualmente não existe nenhuma forma do usuário saber qual padrão os modelos devem seguir, sendo assim um grande obstáculo no trabalho de inserção de novos modelos [Ferreira, 2017]. Um *framework* bem definido, explicando as regras de entrada e saída conseguiria mitigar este problema.

A ferramenta não está concluída e carece de melhorias, que vão desde criação da documentação das regras internas para carregar modelos à implementação de módulos que permitam manusear documentação dos modelos, executar simulações simultâneas, analisar resultados e melhoria da biblioteca de modelos.

2.6 AGWA

AGWA (Automated Geographical Watershed Assesment) é uma ferramenta de modelagem e simulação de *Watersheds*. A ferramenta foi desenvolvida pelo USDA-ARS e funciona como um plugin para o software ArcGIS. Tendo os dados geográficos de uma dada região, normalmente fornecidos pelo próprio USDA, é possível tratá-los dentro da própria ferramenta seguindo um passo a passo que ela impõe à pessoa. Depois de preparados os dados, o usuário pode escolher qual modelo deseja rodar sobre seus dados. Uma vez executado o modelo, os resultados são apresentados na interface do ArcGis por meio de gráficos e alterações nas camadas carregadas. Ainda que a ferramenta forneça as funcionalidades necessário para tratamento dos dados de entrada, é neces-

sário conhecimento de básico a mediano de SIG para entender certas transformações e porque são realizadas [Burns et al., 2004].

O site oficial ¹ conta com tutoriais para alguns conjuntos de dados exemplo. Nesses tutoriais é possível ter uma noção do conhecimento necessário para operar a ferramenta. *AGWA* oferece um bom suporte aos seus usuários pois tem uma interface amigável e bem estruturada. O usuário pode escolher entre seis diferentes modelos na ferramenta [Goodrich et al., 2011]. Os modelos matemáticos não são detalhados nem modificáveis na ferramenta, somente o nome deles é citado, sendo necessário buscar em outras fontes a explicação do modelo e suas particularidades.

AGWA não é uma ferramenta de modelagem e simulação de epidemias, mas conta com os elementos básicos (entrada, tratamento, parametrização, simulação e resultados) de que uma ferramenta desse tipo necessita. A figura 2.3 mostra esses elementos dentro da ferramenta. *AGWA* também é um exemplo de ferramenta acoplada diretamente a um sistema SIG, auxiliando na integração de dados geográficos a modelos [Burns et al., 2004]. Um dos limitantes da ferramenta é a necessidade do uso do *ArcGIS*, que é um software proprietário. Sendo assim, o usuário precisa pagar pela licença do *ArcGIS* para poder utilizar a ferramenta *AGWA*. Além disso, o pré-requisito de conhecimento de algumas funções do *ArcGIS* que estão implantadas em *AGWA* pode ser um problema para usuários mais básicos. Os resultados podem ser manipulados após a simulação, mas internamente não são oferecidos muitos tipos de saídas de dados. A falta de opção de inclusão de novos modelos, produzidos pelo próprio modelador ou encontrados na literatura, também é um limitante.

2.7 Modelo Armadilha de Mosquito

Esta seção tem como objetivo descrever um modelo que precisa de uma ferramenta para que seja possível extrair melhores resultados, acelerando testes e análises.

Lana et al. [2018] descrevem um modelo matemático para a ecologia do mosquito *Aedes aegypti* que faz uso de dados espaço-temporais. O desenvolvimento do mosquito é modulado pela temperatura, via equação de cinética enzimática proposta por Schofield et al. [1981] e parametrizada para o *Aedes aegypti* por Focks et al. [1993]. Para demonstrar a viabilidade desse enfoque, séries temporais de temperatura em bairros do município de Vitória, Espírito Santo, foram obtidas por imagens de satélite e posteriormente tratadas para o modelo. O modelo estimou a capacidade de suporte do ambiente por bairro por meio do ajuste dos dados de infestação por mosquitos obtidos

¹Site oficial - <https://www.tucson.ars.ag.gov/agwa/>

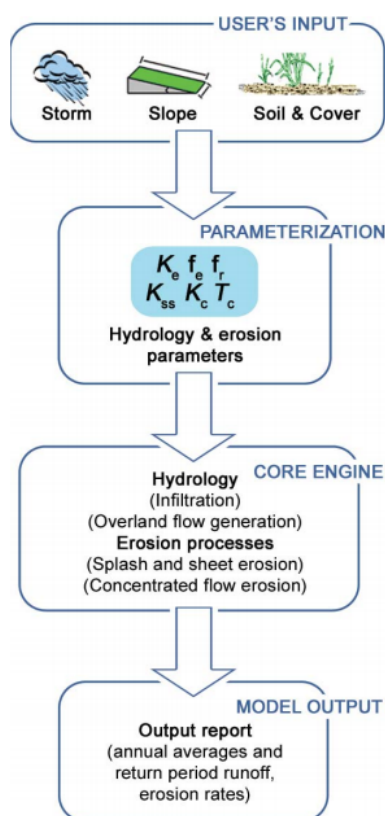


Figura 2.3. Fluxo do AGWA. Fonte: [Burns et al., 2004]

pelo monitoramento de armadilhas. Nessa estimativa utilizou-se também o número de domicílios por bairro e a taxa de captura por armadilha.

Dados de acúmulo de lixo no entorno, presença de bueiros e outras variáveis socioambientais por bairro foram comparadas às estimativas de capacidade de suporte ambiental do modelo, mostrando a correlação entre as condições ambientais e a infestação do mosquito. Estas variáveis, no entanto, não foram incorporadas ao modelo. Sendo assim, é interessante explorar como outros dados podem ser incorporados em um modelo, como por exemplo um indicador de qualidade ambiental, que permita incorporar variáveis como a presença de lixo ou entulho no entorno.

Além disso, outra limitação do modelo é o uso apenas da temperatura como moduladora do desenvolvimento do mosquito. Variáveis importantes como chuva e umidade do ar não foram consideradas, já que a incorporação dessas variáveis aos modelos não é trivial. As dependências espaciais de vizinhança também não foram exploradas nesse contexto.

Este é um exemplo de modelo que precisa de uma ferramenta flexível, já que não seria tão interessante criar um software para manipular somente o modelo, e seria

importante que conseguisse aceitar variações do mesmo. Ao alterar alguma das entradas do modelo, incorporando novos dados, o usuário deveria ser capaz de fazer isso alterando somente o modelo, não o *framework* de software.

2.8 Comparativo

Ferramentas que utilizam dados espaciais e não espaciais para modelos específicos cumprem seu objetivo, mas algumas vezes não oferecem grande flexibilidade ou facilitação nas diversas etapas do processo de modelagem. Modelos mais complexos demandam maior assistência a usuários não familiarizados com ele. As ferramentas *SEARUMS++* e *P2pCodeGeo* contam com um modelo complexo envolvendo dados geográficos, mas que carecem de uma documentação interna do modelo ou apresentação mais explícita de suas variáveis e suas implicações. Além disso, nenhuma das duas ferramentas permite a modificação das regras do modelo pelo usuário, impedindo que o mesmo insira novas variáveis geográficas e espaciais, regras ou tipos de saídas de dados ao modelo.

Neste contexto, vemos que um dos limitantes mais recorrente entre as ferramentas é a falta de diversidade de modelos e entradas que o usuário pode utilizar. *AGWA* e *MalariaTools* oferecem modelos embutidos que podem ser parametrizados e executados pela ferramenta, mas impede que o mesmo seja modificado. Um modelador familiarizado com a linguagem de programação utilizada no modelo deveria ser capaz de modificar o modelo da forma que o melhor atenda, seja identificando melhorias por meio de resultados prévios ou por observações pessoais do fenômeno em estudo. Por outro lado, a ferramenta *Netlogo* não sofre deste problema, pois além de oferecer uma quantidade grande de modelos prontos, é possível programar seus próprios modelos com a linguagem LOGO. Ainda assim, todos os modelos estão limitados ao paradigma suportado pela linguagem, que, apesar de permitir o usuário modificar os aspectos do modelo, não permite criar modelos que não sejam baseados em agentes Tisue & Wilensky [2004].

O *framework* deve visar flexibilidade de linguagens de programação e paradigmas de modelagem suportados, para que assim o usuário possa utilizar de diversas ferramentas disponíveis a ele. Uma biblioteca de modelos e documentação embutida aos modelos ajudam usuários não familiarizados a conhecer melhor os modelos e seu funcionamento. Dentre as ferramentas listadas, somente o *Netlogo* conta com uma biblioteca documentada de modelos para auxílio dos modeladores.

A montagem da interface de parametrização é apresentada no *Netlogo* e *DengueME*, permitindo que o usuário modifique livremente a interface de entrada e saída

dos dados do modelo. Para o *NetLogo*, não é possível a utilização de dados geográficos ou espaciais sem algum tratamento específico. Por exemplo, é impossível imputar uma série temporal de uma planilha do Excel no *Netlogo*. Já no caso do *DengueME*, a ferramenta suporta um leque maior de entradas, mas ainda carece da integração com ferramental que permita a utilização de dados geográficos (como bancos geográficos). O módulo de entrada de dados deve ser flexível o suficiente para aceitar uma certa variedade de tipos de entradas, aumentando assim a gama de opções de modelos que o usuário teria para criar e utilizar. Desde modelos básicos que utilizam somente dados discretos àqueles que envolvam dados provindos de banco de dados geográficos ou planilhas contendo séries temporais deveriam ser possíveis entradas.

A saída do modelo, pós-simulação, é uma das partes mais importantes do processo de modelagem, pois é através dos resultados obtidos que as próximas decisões serão tomadas. Permitir que o usuário exporte dados e defina no modelo ou na interface como estes dados serão exportados ajuda no trabalho de calibração e avaliação dos resultados obtidos. A execução de diversas simulações simultâneas ou de diversos modelos simultâneos, com variações de alguns parâmetros da entrada, não é apresentada em nenhuma das ferramentas aqui citadas. Tendo um módulo de simulação paralela, o usuário poderia configurar diversas rodadas do mesmo modelo ou de variantes dele, com configurações de dados de entradas diferentes. A saída neste caso poderia ser individual ou um mesclado comparativo dos resultados obtidos em todas rodadas, separados por modelos e conjunto de entradas. A tabela 2.1 sumariza as características das ferramentas comparando com o *framework* aqui proposto. Algumas das *features* apresentadas aqui não estavam presentes em nenhuma ferramenta, como a calibragem, comparador de resultados e batch de simulação, estas foram adicionadas baseado na visão do autor sobre o que é interessante se ter em um ferramental de modelagem. A calibragem é uma parte do processo de modelagem para melhoria do modelo e deveria ser incluída. O comparador de resultados e batch de simulação trabalham juntos para melhorar as análises feitas pelo autor e acelerar o processo de finalização do modelo.

Como já explicado, o objetivo do *framework* é servir de base para criação de ferramental de modelagem e simulação, com focos diversos, não somente arboviroses, que consiga auxiliar o usuário em diversos aspectos.

2.9 Considerações Adicionais

Ao checar a literatura encontramos diversos exemplos de modelos e ferramentas de modelagem e simulação para arboviroses e outros campos de estudos. Todas estas fer-

ramentas servem a um objetivo mais específico e atendem aquilo a que se propuseram. A falta de flexibilidade e *features* disponíveis nas ferramentas é um limitante que pode ser resolvido.

A integração de dados geográficos a entrada dos modelos e das ferramentas é um problema recorrente, como demonstrado anteriormente. Dados geográficos e espaciais tem um impacto positivo em modelos. Ao mesmo tempo que trazem maior complexidade ao modelo também aumentam a proximidade dos resultados a realidade. A partir do momento que a ferramenta de modelagem e simulação suporta modelos que utilizam destas entradas, facilitando a integração da entrada e parametrização do modelo, o tempo gasto pelo modelador com outras tarefas é reduzido.

Por fim, as ferramentas aqui apresentadas foram escolhidas baseadas na similaridade delas com o que o *framework* propõe e no fato de cada uma delas tratar de diferentes aspectos do processo de modelagem e simulação. Algumas delas, como o NETLOGO e Malaria Tools, são bem difundidas na comunidade de modelagem e simulação e não poderiam deixar de ser incluídas.

		Ferramentas						
		SEARUMS++	P2PCoDiGeosim	Malaria Tools	NetLogo	DengueME	AGWA	Framework
Features	Entrada Flexível				✓	✓		✓
	Múltiplos Modelos				✓	✓		✓
	Múltiplos Paradigmas de Modelagem					✓		✓
	Múltiplas Plataformas de Modelagem					✓		✓
	Calibragem							✓
	Comparador de Resultados							✓
	Montagem de Interface				✓	✓		✓
	Biblioteca de Modelos				✓	✓		✓
	Integração com Dados Geográficos	✓	✓	✓		✓		✓
	Batch de Simulação							✓
	Edição de Documentação				✓			✓
	Edição do Fonte do modelo				✓			✓

Tabela 2.1. *Features* por ferramenta

Capítulo 3

Metodologia

Neste capítulo é apresentada a metodologia do trabalho, os conceitos envolvidos na criação, as especificações do *framework*, os casos de uso para diferentes tipos de modeladores e os requisitos tecnológicos para construção de uma ferramenta baseada no *framework*.

3.1 Definições e Requisitos

Um *framework* conceitual é uma formulação para criação de aplicações ¹, que contém uma coleção de elementos previamente concebidos com o propósito de facilitar o desenvolvimento de uma aplicação. O capítulo anterior apresentou e comparou ferramentas de modelagem e simulação, cada qual com suas características e recursos (ver Tabela 2.1), criadas para cumprir propósitos específicos. O objetivo deste trabalho é definir um *framework* que englobe a grande variação de estilos e temas cobertos pelas ferramentas apresentadas, oferecendo uma forma de guiar a criação de novas ferramentas de modelagem e simulação tanto estruturalmente quanto logicamente. Uma meta para *frameworks* como o proposto seria a resolução de parte importante da complexidade da implementação de um modelo para simulação, de modo a fazer com que nem todo usuário tenha que ser também um especialista na codificação dos modelos.

O *framework* aqui definido é dividido em nove módulos, cobrindo toda a funcionalidade descrita na Tabela 2.1. Cada módulo tem seu objetivo no apoio ao processo de modelagem e simulação. Além da idealização dos módulos, foram levados em conta os requisitos funcionais necessários para o funcionamento de alguns deles (nem todos precisam de requisitos mais especializados, como paralelização). Os requisitos são apresentados na seção 3.4.

¹<https://techterms.com/definition/framework>

Ao realizar a criação de novos modelos, o modelador implementa um processo ou ciclo de modelagem e simulação. Este processo ou ciclo pode variar para cada caso, mas o *framework* deve ser genérico o suficiente para auxiliar mesmo perante alterações. Além disso, o *framework* pode ser reutilizado de diferentes maneiras, devido à sua especificação mais genérica, podendo ser base de ferramentas mais específicas. Sua arquitetura em módulos pretende oferecer mais flexibilidade ao modelador e ao desenvolvedor. O fluxo de uso é ditado pela forma a qual o *framework* foi modelado, como se fosse um passo a passo ao modelador, visando guiar suas ações para alcançar algum objetivo.

Com o *framework* definido, uma das possibilidades de seu uso é a implementação dos módulos de forma genérica ou específica, permitindo que sejam expandidos ou modificados para atender a necessidades mais específicas [Riehle, 2000]. Para exemplificar a aplicação do *framework*, escolhemos a ferramenta DengueME. Sendo DengueME um software construído originalmente para ser flexível e específico para problemas relacionados à epidemiologia da Dengue, ele oferece parte das funcionalidades descritas neste trabalho. Além disso, DengueME pode ser expandido para comportar funcionalidades que não estejam originalmente incluídas nele.

Os seguintes passos foram seguidos para se realizar a modelagem do *framework*:

1. Revisão bibliográfica em busca de ferramentas de modelagem e simulação, para levantamento de características.
2. Levantamento dos requisitos conceituais do *framework*.
3. Levantamento dos requisitos funcionais.
4. Análise do fluxo de uso para diferentes usuários.
5. Verificação do estado atual da ferramenta DengueME.
6. Teste utilizando DengueME, simulando as partes ainda não construídas na ferramenta.

O principal objetivo da revisão bibliográfica foi ter uma visão das ferramentas atuais e das necessidades dos modeladores. Neste ponto, foi necessário encontrar *gaps* que pudessem ser preenchidos pelo *framework*. Estes *gaps* foram apresentados no Capítulo 2 deste trabalho, na tabela 2.1. Além disso, foi possível encontrar na literatura avaliações sobre o impacto positivo de SIG, ferramentas de modelagem, uso de dados geográficos em modelos e uso de modelos computacionais no geral, corroborando assim com a elaboração de uma base para criação de novas ferramentas. Diversos trabalhos

observam a necessidade de melhoria do ferramental de modelagem e simulação, realizando uma maior inclusão de dados geográficos e oferecendo mais suporte ao modelador por meio de diferentes funcionalidades, tais como um analisador de resultados ou um parametrizador [Wilder-Smith et al., 2017b; Delgado-Petrocelli et al., 2004; Bui et al., 2018; Dom et al., 2017].

Neste contexto, o *framework* precisa ter algumas características chave: **Extensibilidade e Modularidade**, **Flexibilidade** para criação de modelos, **Variedade** de linguagens e plataformas de modelagem, **Customização** de interface, documentação e modelo.

Extensibilidade e Modularidade. O *framework* deve ser construído em módulos, onde cada módulo realiza uma operação que gera uma saída para outros módulos ou salva alguma configuração/dado em disco para acesso posterior. Os módulos devem ser partes completas que possam ser modificadas e estendidas de maneira separada, sem gerar grande impacto em outros módulos. Sendo assim, a ferramenta que segue a arquitetura do *framework* permite que atualizações ocorram sem que exista a necessidade de revisão de funções que já estejam em pleno funcionamento. Por exemplo, ao realizar a inclusão de novos elementos de tela para a montagem da interface de parametrização, estes não devem afetar ou invalidar as interfaces já montadas. No caso da remoção de algum elemento, o modelador deve ser avisado e o construtor de interfaces deve permitir que o modelador ajuste esta lacuna.

Flexibilidade. Uma estratégia comum entre ferramentas de modelagem é a de oferecer um único modelo, embutido no código-fonte da ferramenta, e que não permite qualquer alteração externa. Esse modelo conta com uma interface de parametrização e algumas saídas padrão. Em oposição a essa prática, o *framework* deve oferecer flexibilidade em relação aos modelos que o modelador pode utilizar e/ou inserir na ferramenta. Deve existir a possibilidade de inserir novos modelos, modificar ou utilizar os já existentes na biblioteca da ferramenta. Para tal, é necessário permitir a inclusão de diferentes tipos de modelos por meio de algum mecanismo. Como existe uma grande variedade de modelos, tal requisito viabiliza e justifica a criação de uma biblioteca de modelos que inclua desde modelos simplificados (*Toy Models*) para treinamento de modeladores até modelos complexos, que descrevam fenômenos reais de diferentes áreas. Desta forma, a ferramenta se torna útil para pesquisadores de diversas áreas, assim como é o NetLOGO.

Um dos diferenciais do *framework* proposto em relação ao NetLOGO é a possibilidade de incluir modelos de diferentes paradigmas de modelagem. A ferramenta NetLOGO suporta apenas modelos que utilizem o paradigma de modelagem de agentes. Sabendo da existência de outros paradigmas de modelagem (Dinâmica de Sistemas, Re-

des, Autômato Celular, etc.) e que um mesmo fenômeno pode ser reproduzido nesses diferentes paradigmas, explorando diferentes aspectos ou diferentes variáveis, é interessante oferecer ao modelador o poder de executar modelos de diferentes paradigmas.

Variabilidade. Aceitar modelos escritos em diferentes linguagens de programação é outra característica necessária ao bom funcionamento do *framework*, além de ser um grande desafio. Modelos podem ser programados em diferentes linguagens ou plataformas de modelagem por diferentes motivos: bibliotecas específicas que atendam as necessidades de um dado modelo; modelos antigos já programados naquela linguagem; linguagens de domínio específico (e.g TerraML); necessidade de paradigmas de modelagem variados; e outros. Alguns modelos podem ser reproduzidos em diferentes linguagens. Por exemplo, modelos matemáticos de transmissão (SIR, SI, SEIR) podem ser replicados em qualquer linguagem de programação. Já outros modelos, como por exemplo modelos baseados em redes, não podem ser reproduzidos na linguagem Logo (NetLOGO), pois a linguagem foi feita para modelos baseados em agentes e não comporta a estrutura de uma rede complexa.

Sendo assim, permitir a inclusão de modelos desenvolvidos em diferentes linguagens de programação abre um leque de possibilidades para o usuário, dando mais liberdade para que o mesmo utilize aquilo que melhor se encaixa em seu objeto de estudo. Além disso, não foi encontrada na literatura nenhuma ferramenta pronta que permita o acoplamento de diferentes compiladores-interpretadores, de forma que variados modelos possam ser parametrizados e simulados. A ferramenta mais próxima deste conceito é o DengueME, que atualmente já suporta o TerraME e R, mas carece de melhores mecanismos para a inclusão de mais opções. Para que esta inclusão funcione, a ferramenta precisa ser responsável pela injeção de dados no modelo. Essa injeção se dá por meio da geração de um arquivo que interage com o fonte do modelo e passa a ele todas as entradas. Desta forma, o modelador não precisa se preocupar em definir como seus dados irão para o modelo. Por outro lado, o modelador que deseja inserir um modelo autoral na ferramenta precisa construir o código seguindo regras definidas pelo software montado a partir do *framework*. A exemplo do DengueME, para que os modelos em TerraML funcionem, é necessário que todas as variáveis de entrada e configurações das saídas sejam globais e recebam seus valores de uma variável de nome equivalente. Ao executar um novo modelo, um trecho de código contendo o nome de cada variável e seu valor de entrada pode ser gerado e passado para o fonte do modelo.

Customização. Assim como a ferramenta *NetLOGO* oferece customização da interface de parametrização do modelo, o *framework* precisa contar com um módulo para a construção de interfaces de parametrização para os modelos. O construtor de interfaces de parametrização deve ser simples ao ponto de permitir que modeladores sem

qualquer contato prévio com programação consigam modificar as interfaces dos modelos prontos, seguindo regras impostas pelo *framework*. É necessário oferecer suporte a diferentes tipos de entradas de dados, inclusive tipos mais complexos – como séries temporais e bancos de dados geográficos (definindo os dados da conexão direta com o banco ou um processo de carregamento de arquivos resultantes de consultas no banco). O construtor deve demonstrar ao usuário quais são os meios de ligar a interface ao fonte do modelo, respeitando as entradas que são esperadas pelo código para que ele possa ser executado de maneira correta.

Por fim, a saída é uma das partes mais importantes do modelo e deve receber um tratamento especial. Os dados gerados pela simulação servem de instrumento de estudo do fenômeno ou para melhoria/calibração do modelo. Oferecer diferentes tipos de saídas, gráficas e textuais é desejável e esperado. Cada apresentação dos dados tem um objetivo diferente. A apresentação é bastante importante, e pode ser o diferencial entre um resultado correto e fácil de se entender e um correto, mas confuso [Longley et al., 2005]. Além da apresentação da saída, é necessário oferecer meios para se comparar resultados ou fazer análises sobre os dados gerados (como médias e outras análises estatísticas), dando ao modelador a oportunidade de identificar pontos de melhoria em seu modelo.

Para demonstrar o *framework* na prática, é necessário criar uma ferramenta que o implemente, mesmo que em partes, possibilitando assim a demonstração de como seria a atuação de cada módulo e quais seus impactos. Como informado, a ferramenta escolhida para esta tarefa foi DengueME. A mesma foi escolhida como base de estudos de caso por alguns motivos: (i) A ferramenta foi criada pensando em oferecer flexibilidade para os modeladores, além de ter módulos como o gerador de interface de parametrização já implementados; (ii) DengueME tem bastante espaço para se expandir e para ser refinado, como descrito na revisão bibliográfica; (iii) A ferramenta é funcional em seu atual estado;

DengueME foi concebido como um *framework*, mas nunca chegou a ter seu *framework* formalmente modelado e definido. Os módulos do DengueME foram baseados nas necessidades que um modelador e não modelador têm ao utilizar ferramentas de modelagem, e checadas por meio de contato com ambos os tipos de usuário [de Lima et al., 2016]. Mesmo a ferramenta estando em sua versão 1.0 e disponível online, é possível identificar a necessidade de melhorias em seus módulos, refinando a arquitetura da ferramenta, implementando os módulos aqui descritos não existentes nela, e a tornando uma ferramenta mais completa.

A partir dessa visão do que existe atualmente ao checar a literatura, e tendo descrito as características desejadas para um *framework* que sirva de base para criação

de ferramentas de modelagem e simulação, visando oferecer ao usuário uma maior flexibilidade e facilidade no trabalho, a próxima seção trata da concepção formal do *framework*.

3.2 Projeto do Framework

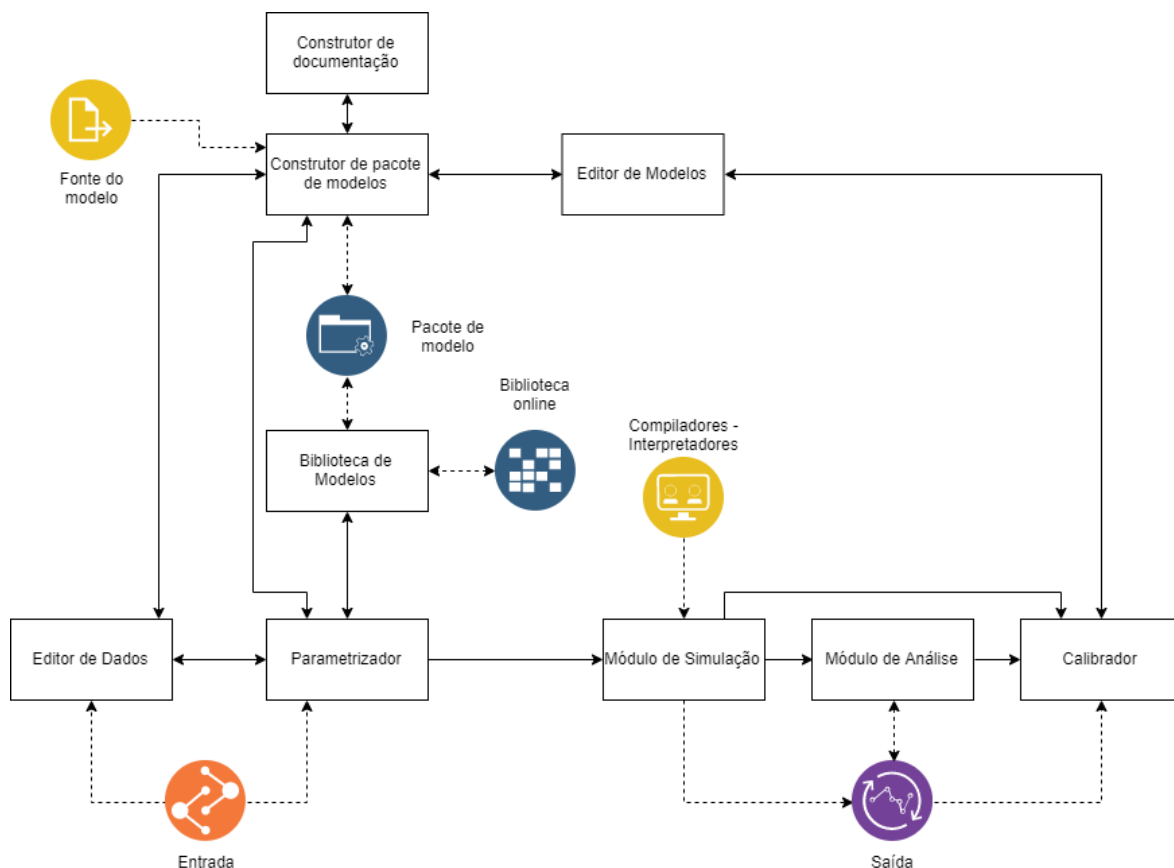


Figura 3.1. Framework

A tarefa de modelagem é complexa [Basu & Andrews, 2013]. A necessidade de se preocupar com a programação do modelo a torna mais difícil ainda. As ferramentas existentes ajudam o modelador, mas podem ser melhoradas ou concebidas de uma forma mais flexível. O objetivo do *framework* é cobrir diversos pontos do processo de modelagem e dar suporte a tipos de dados mais complexos, como dados geográficos, oferecendo assim melhor suporte ao usuário na tarefa de parametrização, análise, simulação e melhoria de seu modelo.

A Figura 3.1 apresenta a concepção geral do *framework* proposto. As caixas representam os módulos do *framework* (um total de 9), enquanto os círculos representam

Por meio do editor de dados, o modelador também pode ter uma ideia de quais entradas são suportadas pela ferramenta, caso seu objetivo seja criar um modelo que esteja disponível na ferramenta para que ele e outros o utilizem.

2. Implementação do Modelo (C): Dentre os módulos descritos existe um editor de modelo (6), que serve para editar o código-fonte na linguagem desejada. Se o modelador desejar, ele pode utilizar o editor para implementar o fonte do modelo. Aqui o usuário terá uma noção das plataformas e linguagens suportadas pela ferramenta.
3. Modelo Computacional (D): Nesta etapa o modelo está pronto ou em fase final, e já pode ser inserido na ferramenta. Os componentes Construtor de pacote de modelos (4), Construtor de documentação (5), Editor de Dados (1) e Editor de Modelos (6) fazem parte da criação de um pacote de modelos – que será explicado adiante – para a ferramenta. O Construtor de pacote de modelos tem relação direta com o modelo computacional. É nele que é criada a interface de parametrização, que está ligada diretamente ao código-fonte. Aqui também são definidos os parâmetros para formatação da saída do modelo. O Construtor de documentação auxilia o modelador a documentar como o modelo funciona, quais suas regras, suas entradas e qual o fenômeno ali explicitado. O editor de dados deve ser configurado para auxiliar potenciais usuários do modelo em como formatar suas entradas. O uso deste módulo é totalmente opcional, uma vez que alguns modelos podem ter entradas de somente taxas ou dados crus, sobre os quais não existe necessidade de se realizar operações. Por fim, o editor de modelos permite que de dentro da ferramenta o modelador faça qualquer modificação ao fonte, seja durante a geração do pacote de modelos, antes ou após ela.
4. Experimentação (E): Durante a experimentação, o usuário deve fazer uso do Parametrizador (2) da ferramenta para criar um novo projeto contendo um modelo da biblioteca ou seu próprio modelo. Ao fazer isso, a interface de parametrização daquele modelo em particular é apresentada, permitindo que o usuário entre com seus dados e parametrize o modelo. Nessa etapa também é possível fazer uso do Editor de Dados (1), caso necessite de alguma operação ou checar que tipo de dado certa entrada no modelo utiliza. Tendo tudo parametrizado, o usuário pode iniciar a simulação do modelo no módulo de simulação (7), onde ele deve escolher coisas como local onde a saída será armazenada (caso exista saída exportável), quantidade de vezes que a simulação deve ser executada, e se deseja

executar simulações do mesmo modelo com diferentes parâmetros em paralelo (se possível).

5. Solução (F): O módulo de Simulação (7) serve para auxiliar o modelador com a solução de seu modelo. Define com qual interpretador/compilador o modelo será simulado (baseado no que foi previamente configurado no pacote de modelos) e se o usuário deseja realizar um batch de simulações com diferentes entradas ou instâncias do modelo.
6. Análise (G): Ao final da simulação, o módulo de análise (8) captura as saídas e os resultados do modelo e oferece ferramentas para análise estatística dos dados numéricos ou comparativos dos resultados, além de conter informações sobre o desempenho computacional do modelo (informação importante para modelos grandes e custosos). Tendo isso feito, o modelador pode optar por realizar alguma calibração em seu modelo no Calibrador (9), seja nas constantes do modelo ou nas regras internas, identificado o que é necessário calibrar (por meio de teste de estresse do modelo por exemplo) o usuário pode usar o editor de modelos (6) para modificar suas regras, sem que isso afete a interface de parametrização ou outras partes.

A associação entre o *framework* e o ciclo de modelagem demonstra o impacto que ele pode ter nas diversas etapas do processo de modelagem e como ele poderia auxiliar modeladores. Na próxima seção os módulos são explicados de forma mais detalhada, visando caracterizar suas funcionalidades mais a fundo.

Cada módulo do *framework* consiste em uma *feature* completa que pode ser utilizada pelo modelador e ou modificada pelo desenvolvedor da ferramenta. Assim se gera um grau de independência entre os módulos, de tal forma que eles possam ser modificados sem que afetem outras partes. Além da análise conceitual do *framework*, é necessário identificar quais são seus requisitos funcionais, para que assim possa se garantir que o *framework* pode ser replicado, expandido, implementado ou modificado. Os requisitos funcionais foram identificados com base nas necessidades do *framework* e no que foi utilizado para desenvolver DengueME.

O fluxo de uso do *framework* para diferentes tipos de modeladores foi analisado visando mostrar o uso da ferramenta. O objetivo é dar assistência e flexibilidade ao modelador.

3.2.1 Módulos do Framework

Nesta seção são apresentados e descritos cada um dos módulos do *framework*.

1 - Editor de dados: O editor de dados permite que o modelador manipule dados de dentro da própria ferramenta, sem a necessidade de instalação de outros softwares e de investir tempo aprendendo como usar cada um deles.

Operações básicas sobre planilhas, arquivos CSV ou até mesmo dados geográficos podem ajudar o modelador a refinar seus dados antes de utilizar os mesmos no modelo. Ao se criar um novo modelo no "Construtor de Pacotes de Modelos", o modelador precisa montar um editor de dados para seu modelo. Esta montagem consiste em definir quais operações são importantes para os tipos de dados que seu modelo espera. Escolhendo dentre as operações disponibilizadas na ferramenta, seria possível indicar quais delas são úteis para manipular os dados como um pré-processamento. Além disso, parte do editor de texto seria dedicada a conter um pequeno passo a passo textual, ensinando o modelador como realizar tais transformações de dentro da ferramenta, quais dados são necessários e como os dados devem estar dispostos. Desta forma, o usuário que nunca teve contato com o modelo sabe tudo que ele precisa para parametrizar o modelo e quais dados ele deveria buscar.

Este módulo é de grande ajuda para usuários não familiarizados com modelagem e simulação ou com o modelo em si. Como demonstrado na revisão bibliográfica, algumas ferramentas não explicam bem suas entradas ou limitam os usuários a usarem somente parâmetros pré-programados na própria ferramenta. Oferecer uma flexibilidade sobre quais dados podem ser utilizados cria uma demanda de auxílio na manipulação destes dados, de forma a não complicar mais ainda o trabalho. Alguns modeladores conhecem as regras envolvidas em dado modelo, mas não entendem a parte computacional da manipulação dos dados de entrada. Tendo um pequeno guia e as operações disponíveis para seu uso, haveria uma redução do esforço necessário para se utilizar um modelo específico, permitindo que o usuário foque em parametrizar vários modelos em um espaço menor de tempo.

O editor é configurado pelo construtor de pacotes de modelos, recebendo dele um arquivo em XML por exemplo, que define as operações e interface do editor. Ao utilizar o editor de dados, o usuário pode optar por enviar os dados editados diretamente para as variáveis correspondentes a ele no parametrizador, da mesma forma, todos dados configurados no parametrizador são demonstrados aqui para possíveis operações ou alterações. O editor de dados pode ser visto como um espelho do parametrizador, mas, diferentemente dele, permite manipular os dados de dentro da ferramenta.

2 - Parametrizador: Pode ser visto como o módulo central, é onde os modelos são parametrizados e projetos contendo os modelos podem ser criados, além de oferecer acesso a todas as outras ferramentas do *framework*. Este módulo seria a *dashboard* da ferramenta, onde o usuário pode interagir com as interfaces de parametrização dos

modelos, configurações de saída, editores de dados, entre outras funções.

Assim que um projeto contendo um modelo é criado, sua interface de parametrização é mostrada na tela. A interface de parametrização apresentada aqui para cada modelo foi previamente montada pelo desenvolvedor do modelo no módulo "Construtor de pacote de modelos", que será apresentado a seguir. O usuário pode alterar as entradas como quiser e quantas vezes forem necessárias. O parametrizador permite também acesso à documentação do modelo e ao editor de modelos, caso o usuário de-seje acessar o fonte do modelo. Aqui também é realizada a parametrização da saída do modelo. A saída é limitada pelo interpretador/compilador usado na criação do modelo que está sendo parametrizado (Exemplo: R ou TerraME). Cada um oferece um leque de alternativas de apresentação de dados que pode ser configurado no parametrizador, e cabe ao usuário modelador torná-las disponíveis em seu modelo ao criar esse parâmetro. Por exemplo, modelos feitos em TerraME com saída gráfica permitem que o usuário manipule pela interface de parametrização quais variáveis devem ser mostradas no gráfico, qual a cor das barras ou linhas, que tipo de gráfico deve ser apresentado, entre outras opções.

O parametrizador é configurado pelo arquivo de interface de um dado modelo construído no construtor de pacotes de modelos. A interface é carregada aqui para que o usuário possa inserir os dados de entrada e configurações de apresentação da saída. A biblioteca de modelos é ligada ao parametrizador para que todos modelos existentes sejam acessíveis ao modelador, permitindo a criação de quantos projetos o mesmo desejar, cada qual contando com seu pacote de modelos e configuração. O parametrizador pega todos os dados de entrada e configurações da saída e os envia para o módulo de simulação para que sejam transformados em código-fonte que venha a se comunicar com o fonte do modelo em si.

3 - Biblioteca de Modelos: Módulo contendo uma biblioteca com todos os pacotes de modelos disponíveis online para a ferramenta ou com as interfaces de modelo criadas localmente. Ao se criar uma nova interface de modelo no Construtor de interfaces, o modelo e sua interface são automaticamente salvos pela ferramenta, permitindo que o modelador o utilize a qualquer momento que desejar. Se eventualmente for de seu desejo, o modelador pode tornar seu modelo disponível online, utilizando a própria ferramenta.

Para construção da biblioteca de modelos online, que pode ser hospedada em qualquer serviço de nuvem, a linguagem de criação da ferramenta precisa suportar comunicação com esse serviço de nuvem, permitindo o upload e download de pacotes de modelos hospedados na biblioteca. Exemplo: A linguagem QT suporta conexão com serviços como Github, que pode ser usado como um repositório colaborativo de

pacotes de modelos.

A biblioteca de modelos guarda todos os modelos existentes localmente e online. Sua principal tarefa é manter todos os modelos organizados e controlar as versões dos pacotes de modelos. Ela se comunica com o parametrizador, expondo a ele todos os modelos existentes para uso. Dentro do construtor de interfaces é possível acessar a biblioteca para inserir novos modelos ou modificar modelos já existentes.

4 - Construtor de pacote de modelos: Módulo inicial para a criação do pacote de modelos. Para o uso de qualquer novo modelo é necessária a sua montagem dentro do construtor, seguindo todos os passos indicados nele. Aqui é onde o modelador cria e personaliza as telas de parametrização da entrada e saída de seu modelo, além de ter acesso ao construtor de documentações e ao editor de modelos.

Neste módulo também é possível descobrir quais entradas e saídas são suportadas pela ferramenta e se seu modelo atende as restrições para funcionar da forma esperada. As opções são suficientes para modelos matemáticos com ou sem dados geográficos. É necessário conhecer o modelo e o código para criação das telas, uma vez que as telas de parametrização serão uma representação gráfica da entrada do modelo.

Ao se abrir o construtor de pacotes de modelos, a primeira coisa a se fazer é inserir informações básicas dos modelos como nome, descrição, plataforma de modelagem desejada (respeitando os limites da ferramenta, ou seja, as suportadas pela mesma) e carregar os arquivos-fonte do modelo.

Feito isso, o usuário parte para a criação da interface de parametrização da entrada e saída do modelo. Cada tipo de entrada é atrelada a um *widget* de tela, e portanto é necessário oferecer uma variedade de tipos de *widgets*. Exemplo: um dado tipo numérico é ligado a um campo de texto que aceita somente números, uma planilha Excel é ligada a um botão de carregar arquivos do computador, uma conexão a banco de dados geográfico é ligado a um *widget* contendo campos para inserir senha, local, endereço e outros. No caso da interface de saída, o usuário escolhe o tipo de saída desejada é apresentado um *widget* completo para aquele tipo de saída que respeita as configurações suportadas pela plataforma de modelagem escolhida pelo usuário. Por exemplo, se o modelo do usuário tiver um gráfico como saída, basta escolher o *widget* de gráfico e atrelá-lo ao código-fonte.

Cabe ao programador analisar qual a melhor analogia entre dado e *widget* e determinar se ela beneficia o usuário em seu trabalho de inserção do dado ou de *configuração* da saída.

A interface de parametrização criada é ligada ao código-fonte do modelo. Essa ligação é feita por meio do nome das variáveis de entrada e saída no fonte. Ao criar a interface de parametrização, o modelador deve nomear cada parâmetro com o nome

da variável de entrada equivalente no fonte. O modelador deve certificar-se que todas variáveis de entrada e saída de seu modelo sejam acessíveis por arquivos externos (hooks em LUA, por exemplo). O mesmo é necessário para a interface da saída. Feito isso, quando o modelo for executado, um arquivo escrito na linguagem da plataforma de modelagem escolhida será gerado pela ferramenta. Esse arquivo contém todas as configurações das entradas e saídas inseridas antes da simulação. Desta forma, o usuário monta a interface como desejar e a conecta ao seu fonte, respeitando as regras impostas pela ferramenta.

Ao final da montagem da interface de entrada e saída, documentação e parâmetros do editor de dados, um pacote de modelos será gerado. Esse pacote pode ser inserido na **Biblioteca online** ou ser mantido somente na máquina local.

O Construtor se comunica com todos módulos pertinentes ao pacote de modelos - Construtor de Documentação, Editor de dados - e com o módulo "Editor de modelos" para identificar possíveis alterações nas variáveis de entrada e saída do modelo. O código-fonte do modelo é inserido na ferramenta através do construtor de pacotes de modelos. O fonte é guardado dentro do pacote de modelos que será aqui construído e futuramente inserido na biblioteca de modelos.

5 - Construtor de Documentação: Módulo para documentação dos modelos, visando dar descrição a entrada e explicação de como o modelo é implementado, tornando assim transparente a lógica do modelo e eliminando a necessidade de leitura do código-fonte. O construtor de documentação permite a criação de uma página HTML ou um documento PDF que será acoplado a todo o pacote do modelo. Quando o modelo for postado online, a documentação o acompanhará, juntamente de sua interface de parametrização e código-fonte.

Este módulo se comunica somente com o construtor de pacotes de modelos. Não é necessário o mesmo se comunicar com o parametrizador pois uma vez que o pacote de modelo é criado, o parametrizador carrega o arquivo de documentação partindo dele.

6 - Editor de modelos: O editor de modelo permite a edição do código-fonte do modelo direto na ferramenta. Uma vez que o modelador identifique melhorias pelas análises feitas no calibrador ou decida alterar alguma regra do modelo com base em suas observações sobre os resultados, deve ser possível fazer isso diretamente na ferramenta sem causar impacto nas outras instâncias desse modelo. A única parte do fonte que não pode ser alterada sem causar impacto são os nomes e tipos das variáveis de entradas e saídas, pois as mesmas estão diretamente atreladas à interface de parametrização criada.

A alteração da regra do modelo, de programação, ou de constantes em fórmulas ocorre de forma a não afetar todo o resto, economizando tempo de trabalho e tornando

prático o processo de melhoria. Ao se alterar o modelo, uma nova versão dele é salva em disco, e o usuário tem a opção de sobrescrever ou não a versão antiga. Ter várias versões pode ser interessante para futuras comparações de resultados utilizando o batch de simulações.

O editor de modelos está ligado ao calibrador, para que o usuário o acesse enquanto vê as possíveis modificações necessárias apontadas pelo calibrador. Deve ser possível editar o código e recarregá-lo diretamente no calibrador para novos testes.

7 - Módulo de simulação: O módulo de simulação controla o tempo de simulação, onde serão salvas as saídas, a geração do fonte contendo as entradas colocadas no parametrizador e os interpretadores/compiladores necessários ao modelo. A plataforma de exemplo, DengueME, suporta atualmente modelos em R e TerraME. A ferramenta processa todos os dados de entrada, convertendo os mesmos para a linguagem escolhida nas opções de simulação (que devem ser consistentes com o modelo sendo utilizado) e gerando um fonte contendo os dados de entrada. Após esse processamento, o interpretador ou compilador da linguagem é chamado e executa o código (parâmetros de entrada e modelo). Ao final da execução, a saída configurada no parametrizador é apresentada ou armazenada em disco.

A saída é externa à ferramenta, mas faz parte de seu *workflow*, sendo fornecidas alternativas pelos interpretadores mas tendo parâmetros configurados dentro da ferramenta. Ao final da execução de um modelo, as configurações da saída (já incluídas na entrada do modelo) definem que tipo de saída deve ser apresentada. Seja um gráfico ou um arquivo *png* que deve ser salvo em disco, o usuário deve ter entendimento de qual saída escolheu para saber analisá-la. Todas saídas armazenadas podem ter seu caminho escolhido no módulo de simulação.

Aqui se nota um dos limitantes do *framework*, uma vez que não existe uma forma genérica de gerar código-fonte de diferentes interpretadores. É necessário que os modelos sejam escritos naqueles suportados pelo software de modelagem. Para reduzir este limite, o *framework* suporta que modificações em seu módulo de simulações sejam feitas, permitindo a inclusão de novas plataformas de modelagem ou linguagens de programação sempre que necessário, de modo que isso não afete o resto da ferramenta.

É possível configurar no módulo de simulação um procedimento *batch* de simulação, que ajuda a criar um conjunto de casos a serem simulados. O usuário configura quantas vezes deseja rodar um mesmo modelo e, se deseja, usar diferentes entradas para cada uma das rodadas. Ao final, tudo é executado e seus resultados são armazenados em disco para futuras comparações no analisador de resultados.

8 - Módulo de análises: O processo de modelagem e simulação consiste num ciclo, em que diversas revisões são feitas durante a criação de um modelo ou a partir

da análise de casos. O módulo de análises conta com duas ferramentas chave: análises matemáticas dos dados e comparação de resultados.

O comparador de resultados se aproveita do *batch* de simulações, apresentando todos os resultados obtidos nas diversas simulações realizadas (ou carregando resultados de simulações previamente feitas). Além disso, dentro do módulo de análises, uma ferramenta para alguns cálculos básicos deve ser disponibilizada. Oferecendo operações básicas como média, desvio padrão e outros. É possível ofertar uma calculadora científica dentro da própria ferramenta e que automaticamente carregue os dados da solução caso estes sejam numéricos.

Este módulo recebe a saída e as configurações do módulo de simulação (como os dados de entrada e tempo de simulação). O usuário pode também carregar saídas externas de modelos semelhantes ou de execuções anteriores. Se o usuário desejar, ele pode carregar dados reais para comparar com seus resultados, mas isto preferencialmente deveria ser feito através do calibrador, que é configurado para tal, uma vez que o módulo de análises deveria aceitar somente saídas do mesmo tipo do modelo sendo executado agora, saídas em outros formatos podem não funcionar como deveriam. O módulo de análise serve de ponto de partida para o calibrador, que faz uso de resultados conhecidos para analisar os resultados retornados pelo modelo, ajudando o modelador a identificar a corretude de seu modelo.

9 - Calibrador: O calibrador de modelos permite o carregamento de resultados reais para a comparação e calibração do modelagem com base nos resultados apresentados por ele após a simulação, sendo essa uma ferramenta para usuários avançados. O calibrador deve contar com técnicas para calibragem de modelos do tipo que a ferramenta suporta. Por exemplo, se uma ferramenta de modelagem epidemiológica é criada a partir do *framework*, métodos como o adotado pelo trabalho de Kong and Macmahon Kong et al. [2009] podem ser ofertados. Carley Carley [1996] apresenta um fluxograma de como pode ocorrer o processo de calibração de modelos computacionais e demonstra qual a sua importância em um modelo deste tipo.

Existem diversas técnicas de calibração, mas não é possível afirmar qual seja a melhor delas de maneira genérica. Cada modelo e objeto de estudo tem suas próprias técnicas, que devem ser estudadas para que sejam ofertadas na ferramenta [Park & Qi, 2005; Stout et al., 2009]. A calibragem é uma área de estudo especializada e necessita de profissionais específicos para se aprofundar e tomar decisões de quais técnicas usar para cada modelo.

O calibrador é sem dúvida um dos módulos mais complexos deste *framework*, mas de grande utilidade, uma vez que a calibração é uma peça importante para que o modelo atinja os resultados esperados e seja válido para descrever o fenômeno de

estudo [Carley, 1996]. Ele deve receber as saídas do módulo de simulação e arquivos de saída externa que representem algum resultado já conhecido pelo modelador, para que assim possa comparar os resultados de seu modelo com dados reais e checar possíveis diferenças. De dentro do calibrador, o modelador pode abrir o editor de modelos para modificações nas regras do modelo. O calibrador em si não envia nenhuma informação para o editor de modelos (salvo o modelo que está sendo calibrado no momento).

Independência. Devemos lembrar que os módulos tem um grau de independência e alguns deles podem ser utilizados separadamente. Por exemplo, o calibrador pode ser aberto a qualquer momento para qualquer modelo, não necessariamente sendo necessário que uma simulação seja executada. Já o módulo de análises precisa que uma simulação seja previamente feita para que o mesmo carregue o resultado para análise.

Os módulos de criação de documentação e editor de dados podem ser utilizados para qualquer modelo a qualquer momento, seja para checar os dados do mesmo ou realizar alguma modificação no pacote. Os editores de modelos e de interface gráfica podem ser abertos também a qualquer momento para qualquer modelo, permitindo a modificação do fonte e da interface de parametrização.

3.2.2 Outros componentes do framework

Nesta seção são explicados componentes externos ou internos que têm impacto no *framework* (indicados com círculos na Figura 3.1).

Entrada: A entrada consiste no conjunto de dados que será inserido na ferramenta para alimentar o modelo escolhido pelo modelador e assim executar simulações sobre o mesmo. A entrada pode ser um conjunto de constantes ou até mesmo uma conexão a um banco de dados geográficos. É dever da ferramenta oferecer suporte para os mais diversos tipos de fontes de dados, desde que exista um modelo que os utilize. O tipo de dado aceito é limitado pela linguagem utilizada para construir a ferramenta, por exemplo, a linguagem QT contempla os mais diversos tipos de entradas, permitindo que modelos com entradas complexas sejam acoplados à ferramenta.

Note-se que dados brutos nem sempre são suficientes como entrada para um modelo. Por conta disso, o *framework* conta com módulos para auxiliar o modelador na tarefa de transformação, correção, normalização ou seleção dos dados, sendo estes o Parametrizador e o Editor de Dados.

Fonte do modelo: O fonte do modelo é carregado na ferramenta durante a criação do pacote de modelos. O mesmo será associado a uma interface de parametrização customizada pelo modelador, tendo suas variáveis de entrada e saída diretamente liga-

das a ela. O fonte pode ser visualizado e manipulado de dentro da própria ferramenta no editor de modelos.

Biblioteca online: A biblioteca online é uma biblioteca global de modelos que permite que os modeladores incluam seus modelos para que outros usuários da ferramenta os utilizem. De dentro da ferramenta deve ser possível sincronizar a biblioteca online, carregar novos modelos para a mesma ou realizar modificações de seus modelos já online. A ideia é se ter uma biblioteca comunitária, mas que tenha algum mecanismo que impeça que terceiros tirem seus modelos. É necessário envolver um mediador no processo de upload de novos modelos ou de alterações, a fim de manter a integridade da biblioteca.

Compiladores e interpretadores: Para se usar diferentes linguagens é necessário que o modelador tenha seus respectivos compiladores ou interpretadores instalados em sua máquina. Dentro do módulo de simulação o modelador pode checar se todos eles estão configurados. Caso contrário, é ofertado ao modelador um local para instalação do interpretador ou compilador necessário àquele modelo que está sendo parametrizado.

Pacote de modelos: Pacote de modelo é um produto gerado pela ferramenta ao se inserir um novo modelo. Este pacote é criado pelo "Construtor de pacotes de modelos". O pacote de modelos é composto por quatro partes: código-fonte do modelo, descrição da interface de parametrização em XML, um arquivo html ou PDF de documentação, e os parâmetros que definem o editor de dados. A montagem do pacote começa no construtor de pacotes de modelos, depois passa pelo construtor de documentação e pelo editor de dados. A Figura 3.3 ilustra o pacote de modelo que é gerado pelo construtor de pacotes ao final de todas as etapas.

Dentro do pacote, somente o código-fonte e a configuração da interface de parametrização são obrigatórios para o funcionamento. Uma vez que é possível alterar qualquer parte do pacote a qualquer momento, a adição da documentação e os parâmetros do editor de dados não se tornam um problema.

3.3 Fluxo de uso

O *framework* é feito de forma que descreva o fluxo interno da ferramenta e consequentemente permita conceber um fluxo de uso, guiando os usuários de forma a tirarem total proveito do *framework*.

Aqui são apresentados dois exemplos de fluxos de uso, demonstrando do início ao fim quais as etapas esperadas e aconselhadas. O primeiro fluxo é um exemplo de



Figura 3.3. Pacote de modelo.

modeladores que já tiveram contato com programação e já produziram ou utilizaram modelos matemáticos computacionais. O segundo fluxo é voltado para modeladores que nunca tiveram contato com programação mas se interessam em utilizar modelos prontos para seus estudos.

Cabe lembrar que o *framework* não se limita a esses dois tipos de usuários, que são somente os exemplos mais comuns. Qualquer pessoa deveria ser capaz de utilizar o mínimo da ferramenta, a partir da apresentação de seus módulos de forma didática, para que assim atenda a mais pessoas.

3.3.1 Modelador programador

O modelador é aquele que detém conhecimento de dado fenômeno e produz modelos matemáticos que representem o fenômeno de estudo. O modelador programador, além de produzir o modelo matemático, precisa traduzir este modelo em código, utilizando alguma linguagem de programação. Normalmente se escolhe uma linguagem atrelada a uma plataforma de modelagem e simulação, por exemplo, Logo para o NetLOGO e TerraML para o TerraME. Em outros casos é possível usar linguagens de uso geral, como R e Python, que contam com pacotes com ferramental para tal tarefa.

Tendo o modelo em uma primeira versão, funcional, e que utilize alguma plata-



Figura 3.4. Caso de uso.

forma/linguagem suportada pela ferramenta baseada no *framework* o modelador pode começar a utilizá-la para auxiliar e facilitar seu trabalho. A figura 3.4 descreve o caso de uso imaginado para este tipo de modelador.

Ao abrir a ferramenta pela primeira vez, o modelador deve ir até as configurações e certificar-se que o interpretador e/ou compilador utilizado para construção do modelo é suportado, e se ele está configurado corretamente. O modelador deve checar também a linguagem da ferramenta e o caminho até sua biblioteca de modelos.

Uma vez configurada a ferramenta, é necessário ir até o Construtor de pacotes de

modelos para iniciar a inserção do modelo em um pacote de modelo que pode ser lido pela ferramenta, permitindo a parametrização e simulação do mesmo. Aqui será criada a interface que será utilizada na parametrização da entrada e saída do modelo, será gerada uma documentação para o modelo e serão selecionados os parâmetros do editor de dados. O Construtor guia o modelador primeiramente na montagem da interface de parametrização para que uma interface seja montada por completa. Tendo a interface de parametrização construída, as entradas especiais (como configurações de conexão de banco) configuradas e as saídas do modelo definidas o modelador pode iniciar a criação da documentação do modelo. A documentação é gerada em formato html e pdf, estando disponível no *help* do modelo, ou online, caso o modelo seja hospedado na biblioteca de modelos online. A última etapa consiste em montar o editor de dados. O editor de dados serve de auxílio a usuários que não conhecem como deve ser a entrada do modelo. O modelador pode então indicar quais operações devem ser realizadas nos dados (tendo em vista as disponibilizadas pela ferramenta) e em qual ordem as realizar. Tendo tudo isso feito, o modelador pode salvar seu pacote de modelo e ele estará pronto para ser usado diretamente na ferramenta.

Uma vez criado o pacote de modelos, o modelador pode optar por salvar seu modelo na biblioteca de modelos local ou online. Feito isso, agora é possível ir até a *dashboard*, iniciar um novo projeto, abrir o modelo, parametrizar suas entradas e saídas utilizando a interface criada e iniciar a experimentação.

Após execução do modelo, as saídas são mostradas na tela ou salvas na máquina (depende de que tipo de saída o modelo tem). Se o modelador optar, ele pode utilizar o analisador de resultados para tirar o máximo de seus resultados. Caso o modelador deseje melhorar seu modelo ou checar sua validade, ele tem a opção de acessar o calibrador para realizar uma calibração de seu modelo.

3.3.2 Modelador não programador

O modelador não programador é aquele que não tem entendimento em programação, mas tem conhecimento em alguma área de estudo a qual existem modelos matemáticos computacionais que podem ser usados em seus estudos. Supondo que estes modelos já estejam dentro da biblioteca de modelos do *framework*, o modelador terá somente o trabalho de obter os dados necessários para a execução do mesmo. A Figura 3.5 descreve o caso de uso para este modelador.

Uma vez aberto o *framework*, o modelador deve primeiramente configurar a ferramenta, certificando que as plataformas de modelagem necessárias estão instaladas em sua máquina. Em seguida, o modelador deve criar um novo projeto, acessar a biblio-

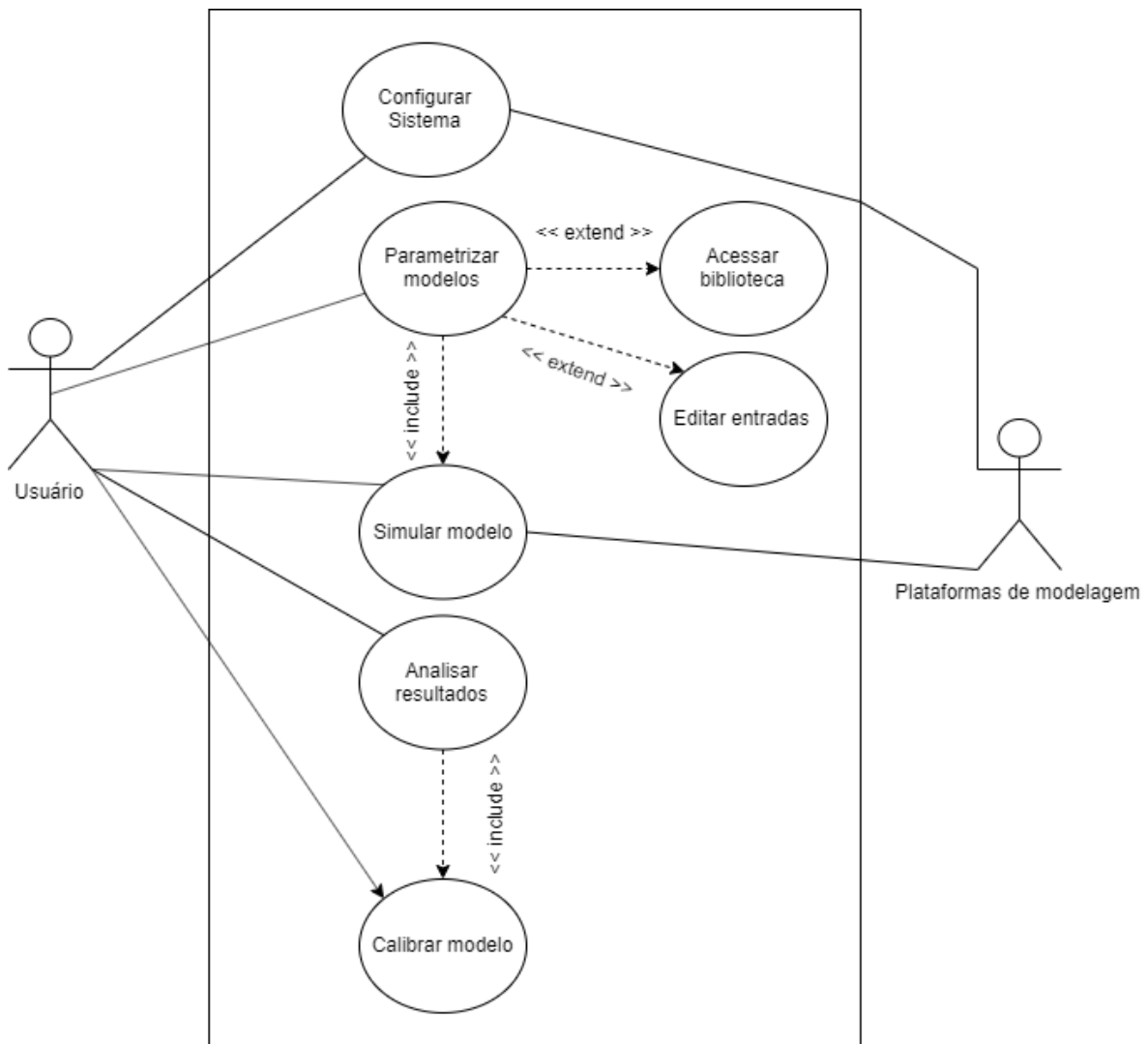


Figura 3.5. Caso de uso.

teca de modelos e escolher qual modelo deseja parametrizar e simular. É apresentada ao modelador uma interface de parametrização, um botão para abertura do editor de dados e um botão com a documentação do modelo.

O modelador deve abrir o editor de dados para se informar sobre quais entradas são aceitas e necessárias no modelo. Aqui também é demonstrado se as entradas puras precisam de alguma transformação ou modificação. Caso precisem, o próprio *framework* oferece meios de manipular os dados, bastando o modelador indicar quais operações deseja fazer, sobre qual tipo de dado sera feito e carregar os dados. Cabe lembrar que o modelador programador que inseriu este modelo na biblioteca de modelos é quem teve o trabalho de definir as entradas e quais as operações necessárias.

Finalizado o tratamento dos dados, o modelador pode voltar à interface de para-

metrização e iniciar a inserção dos dados. Uma vez inseridos todos os dados de entrada, é hora de configurar as saídas dos modelos na interface de parametrização. As saídas são configuradas baseando-se naquilo que o modelo suporta, mas nem sempre é possível oferecer todos tipos de saídas em um mesmo modelo.

Por fim, o modelador pode agora optar por rodar uma simulação única ou utilizar o batch de simulação caso tenha em mente uma análise de resultado utilizando diferentes entradas. Escolhida a forma, a ferramenta se encarrega de injetar todos dados inseridos no modelo. Uma chamada para o interpretador da linguagem escolhida é feita e o modelo é executado.

Dependendo de como são as saídas do modelo as mesmas serão apresentadas na tela ou salvas em disco. Tendo a solução, o modelador pode utilizar o analisador de resultados para fazer comparações, computar médias, calcular desvios ou realizar análises visuais.

Feito todas análises o modelador pode voltar ao parametrizador para mais testes ou utilizar o calibrador para ajustar constantes ou regras do modelo. Essa operação não afeta a interface de parametrização e nem causa problemas com os outros modelos, sendo independente.

3.4 Requisitos funcionais

Aqui são apresentados alguns requisitos necessários para que o funcionamento do *framework* descrito aqui ocorra e a justificativa da preferencia de algumas tecnologias para implementação de um software baseado neste trabalho. Os requisitos apresentados foram levantados baseados na construção do DengueME e na análise do que seria necessário para funcionamento de todos módulos propostos.

A ferramenta exemplo, DengueME, foi construída utilizando o *framework* QT para C++. QT é um *framework* poderoso que conta com diversas bibliotecas e outros softwares de sucesso. QT foi usado na construção do software QGIS, uma ferramenta *open source* de manipulação de dados geográficos que compete com o ArcGIS atualmente.

O *framework* precisa de uma interface gráfica e de um construtor de interfaces gráficas. Ambos podem ser feitos pelo QT, que oferece uma biblioteca completa de widgets de tela e que podem ser utilizados para um construtor de interfaces de parametrização. Os elementos de tela podem ser combinados em widgets mais complexos ou utilizados de maneira singular. Além disso, é possível manipular XML e HTML por meio do QT, permitindo a criação da documentação do modelo e de qualquer outro

arquivo de configuração que venha a ser estruturado em arquivos deste tipo (como, por exemplo, a base dos modelos).

Aqui são apresentados requisitos especiais que alguns módulos precisam:

Editor de dados: Suporte a diversos formatos de arquivos, que possam configurar entradas de dados, como csv ou xls. Suporte a manipulação destes arquivos de entrada e geração de novos arquivos ou dados para uso na entrada dos modelos. Por fim, suporte a operações sobre dados geográficos. QT pode fazer uso da API do QGIS para tais operações. Além disso, dentro da própria API do QGIS, existe a biblioteca *GRASS*, que oferece operações complexas e de mesmo poder que o ArcGIS. Tanto as ferramentas do QGIS (QT e Python) quanto a biblioteca *GRASS* foram avaliadas para que AGWA fosse implementado com base nelas [Ferreira, 2015]. Chegou-se à conclusão que ambas tem tudo necessário para modelagem e simulação [Ferreira, 2015]. Por fim, deve oferecer suporte a diferentes operações sobre diversos tipos de dados.

Biblioteca de modelos: Suporte a conexão web para upload e download dos modelos na biblioteca online. QT possui bibliotecas para comunicação web e que tornam possível essa funcionalidade. Permitindo assim que repositórios em nuvem sejam configurados para guardar os modelos compartilhados.

Módulo de simulação: Suporte a paralelismo e comunicação com o sistema operacional para realizar chamadas de sistema. QT tem suporte para paralelismo, contando com sua própria implementação de *threads* com diversas melhorias e funcionalidades. Outra opção seria executar os modelos em fila, aumentando assim o tempo total de conclusão de todas simulações, mas sem a necessidade de qualquer *feature* específica para ser implementado.

Editor de modelos: Suporte a carregamento e manipulação de arquivos de diferentes linguagens de programação.

Calibrador: Suporte a leitura de arquivos de diferentes tipos que possam conter resultados de outras simulações ou dados que representem a solução de algum caso de estudo conhecido.

Todos módulos precisam de um estudo para serem implementados, e deve ser feito um levantamento quanto ao que é pertinente para o objeto de estudo da ferramenta. Para DengueME, foi iniciada a mudança de sua arquitetura e reengenharia de seus módulos de modo que a ferramenta se adéque ao *framework*. Parte deste trabalho de recriação do DengueME não pode ser concluído durante esta dissertação e são atividades reservadas a trabalhos futuros, mas no capítulo quatro é apresentado todo trabalho feito, em progresso e que ainda está em estágio inicial.

Capítulo 4

Reengenharia e Extensão do DengueME

Como explicado anteriormente, a ferramenta DengueME foi usada como exemplo de software que segue parcialmente o projeto do *framework* proposto. No estado que se encontrava, versão 1.0, diversos requisitos do *framework* não estavam implementados, alguns existiam de maneira incorreta e outros precisavam ser melhorados.

Esta seção apresenta a reengenharia do código do DengueME de modo a torná-lo aderente ao *framework* proposto, e assim servir de base para a implementação completa do *framework*. Após essa reengenharia, temos o equivalente à implementação do *framework* proposto, particularizado em diversos aspectos para usar código do DengueME, mas preparado para futuras expansões visando englobar outros paradigmas de modelagem e ferramentas de apoio. Com o DengueME refatorado, desenvolvemos estudos de caso no Capítulo 5 para demonstrar a validade do *framework* proposto.

A arquitetura do DengueME é representada na 4.1. Esta arquitetura não foi planejada originalmente e é uma consequência do desenvolvimento sob demanda e sem grandes preocupações quanto à sua expansão. Nota-se que a arquitetura original, se comparada ao *framework* tem poucos módulos e não oferece muitas opções para o usuário, oferecendo um fluxo bem simples e centralizado. Pela figura podemos ver que o parametrizador é o centro do sistema, tendo a biblioteca de modelos e construtor de interfaces ligados a ele. A biblioteca tem a única tarefa de organizar localmente os modelos que podem ser parametrizados, enquanto que o construtor de interface permite o usuário a criação de interfaces para seus modelos.

Tendo a interface e o modelo podemos entrar com os dados no sistema. Feito isto, o módulo de simulação que não tem interface gráfica, sendo ele somente um módulo no fonte, realiza o parser para a linguagem da plataforma de modelagem e simulação

para que o modelo seja simulado e o resultado apresentado.

Olhando para o *framework* que descreve DengueME é possível verificar que muito do que é proposto aqui não existe na ferramenta, além disso, o código foi construído de forma a comportar somente estes módulos, dificultando um pouco a extensão. Analisando estes fatos e tendo em mente tudo que já foi apresentado a este ponto do trabalho, podemos dizer que a ferramenta não consegue atingir o nível desejado de flexibilidade, impacto nas atividades do usuário e extensão. Sendo que estes fatores são objetivos da mesma.

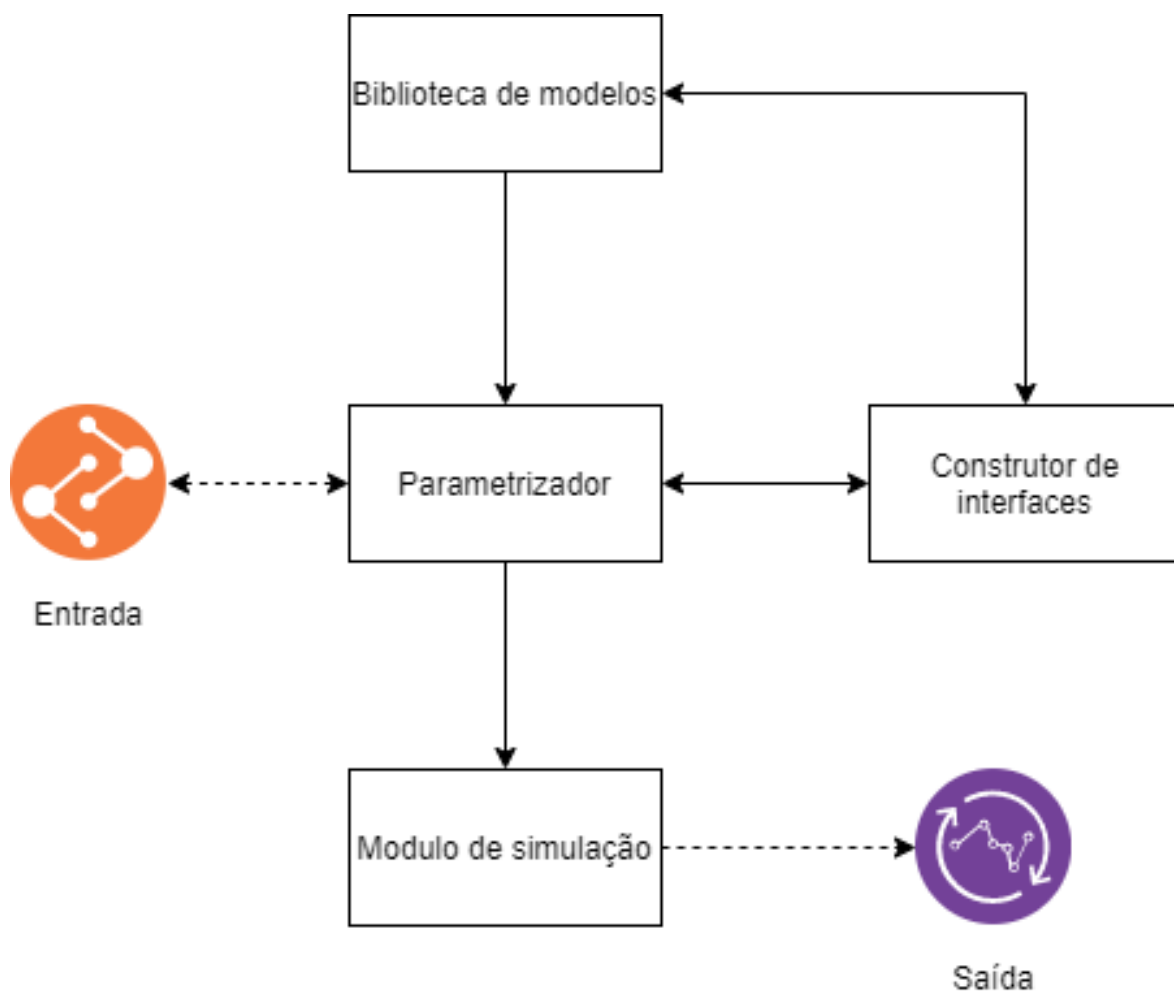


Figura 4.1. Arquitetura DengueME original.

A modificação do DengueME de modo a adequá-lo à proposta de *framework* ocorreu em quatro etapas: (1) reengenharia do fonte para reconstruir partes do código e eliminar elementos não utilizados; (2) Modularização o sistema, para que o mesmo se torne aberto a extensões e melhorias futuras; (3) Início da extensão do sistema, introduzindo módulos faltantes e iniciando o entendimento funcional do necessário para

cada um; (4) Atualização da documentação da ferramenta. Nas seções a seguir, cada uma das etapas é detalhada. Lembrando que durante este trabalho não foram concluídas todas as etapas, uma vez que algumas delas demandam estudos mais aprofundados ou tempo de implementação e teste.

4.1 Reengenharia

O primeiro passo para melhoria do DengueME foi a reengenharia de seu código como um todo. As funcionalidades do DengueME foram originalmente implementadas de forma incremental, por demanda, o que acabou levando a uma arquitetura mais próxima da monolítica e com diversas pendências. Para o *framework*, esse tipo de arquitetura não é o ideal, uma vez que é esperado que o software gerado seja flexível, extensível e modular. Portanto, é necessário ajustar a arquitetura e o código do DengueME.

Para atender aos requisitos necessários, todas as classes foram refatoradas de forma que se agrupem em módulos. Cada tela do sistema pertence agora a algum módulo denominado no *framework* e não mais estão soltas ou ligadas a uma única entidade, como originalmente. As respectivas classes foram alteradas para não afetarem as outras a ponto de quebrar o sistema caso alguma modificação seja realizada. Por exemplo, o construtor de pacotes de modelos e o parametrizador eram ligados um ao outro, uma vez que o construtor seria uma instância “editável” do parametrizador. Isso causava um problema no qual, toda vez que a interface do construtor de pacotes fosse alterada, a do parametrizador sofria consequências. Outra dependência indesejada existente era que os pacotes ainda em construção compartilhavam as mesmas pastas em disco dos modelos já construídos, causando problemas com as versões dos modelos já existentes e impedindo o seu versionamento correto (função essa que ainda não existe no código). Os módulos devem somente se comunicar uns com os outros, não é viável permitir que eles compartilhem classes ou interfaces gráficas uma vez que suas funções são completamente diferentes.

DengueME continha diversos trechos de código antigos e não utilizados que precisavam ser eliminados. Partes obsoletas ou inutilizadas do código foram removidas para evitar problemas e facilitar o uso do código do DengueME como exemplo de uma implementação do *framework*. Algumas bibliotecas desnecessárias foram removidas, como a utilizada para compactar arquivos (zlib), pois o próprio QT oferece uma interface para compactar e descompactar o pacote de modelos.

Outra parte importante da reengenharia foi a do módulo de interpretadores do DengueME. Para se inserir um novo interpretador/compilador em DengueME, é ne-

cessário realizar o mapeamento dos elementos gráficos disponíveis ao usuário à sintaxe da linguagem daquele novo interpretador. Ou seja, se um usuário inserir um campo de texto na interface de parametrização, um trecho de código equivalente a esse campo e o valor que ele irá receber deve ser gravado no arquivo que é gerado ao iniciar a simulação. Este gerador de código-fonte estava em um estado experimental, aceitando somente o TerraME e R como linguagens. O *parser* de cada interpretador se encontra agora separado das outras partes do sistema, permitindo sua livre alteração, sem que ocorra efeito colateral aos *parsers* já implementados. Além disso, uma documentação sobre como criar um *parser* deste tipo foi criada. A nova documentação é tratada adiante.

Em conclusão, a arquitetura, organização e código do DengueME foram refatorados visando o *framework* criado no trabalho. Se o software tivesse sido construído desde o início baseado em algum *framework*, este trabalho de reengenharia não seria necessário. Neste contexto, podemos ver que planejamento é uma parte importante da construção de ferramentas. Ter uma forma de se guiar e de decidir o que implementar agiliza o trabalho e minimiza o retrabalho.

4.2 Modularização

Uma das etapas realizadas foi a de mapear o sistema existente para realizar a modularização. DengueME contava com alguns módulos implementados, mas sem separação ou definição precisa. Para permitir a extensão do sistema é necessário garantir que o código existente execute suas tarefas como modelado e que não cause interferências em outras partes do sistema.

Os seguintes módulos existem no DengueME original: construtor de pacote de modelos, biblioteca de modelos, parametrizador, construtor de documentação. Para cada um deles foi necessário separar suas partes e eliminar ligações desnecessárias. Por exemplo, o DengueME original permite abrir o construtor de documentação diretamente a partir do parametrizador. Esse não é um comportamento esperado pelo *framework*, uma vez que no parametrizador o usuário deve ter acesso à documentação do modelo pronta e não deveria ser capaz de editá-la.

Outro cuidado tomado foi com a separação da interface gráfica. Diversas telas do DengueME faziam uso de um mesmo arquivo de design, que era modificado em tempo de execução pelo fonte considerando a tela aberta no momento. Para manter a modularização, é necessário permitir alterações nas interfaces dos módulos sem que estes prejudiquem outras telas. Logo, cada módulo tem seu próprio conjunto de designs

de telas e mantém as classes que as controlam. DengueME foi desenvolvido usando a biblioteca QT, que por sua vez tem partes já modularizadas, facilitando assim a separação dos módulos dentro do código e o isolamento dos designs de interface de cada tela.

A classe central da aplicação é o parametrizador, sendo deste a primeira tela apresentada ao usuário. Partindo dele deve ser possível acessar as outras telas, que podem por sua vez levar a mais telas internas ao sistema. Esse fluxo não existia em DengueME e precisou ser criado, para que assim a ferramenta dite um passo a passo para seu uso. O novo fluxo seguiu a ideia do *framework*, com cada módulo respeitando suas ligações e oferecendo o acesso aos módulos corretos.

4.3 Extensões e Implementações Pendentes

Uma ferramenta de modelagem baseada no *framework* proposto deve aceitar extensões baseadas em demanda. No caso do DengueME, a primeira extensão que se torna viável a partir de sua adequação ao *framework* é a implementação dos módulos que faltam a DengueME e foram propostos: Editor de Dados, Módulo de Simulação, Módulo de análise, Calibrador e Editor de modelos.

O módulo de simulação existe em parte, mas está acoplado ao parametrizador. O processo de isolamento é um trabalho em andamento, dando a ele uma tela única e suas próprias classes de controle. O parametrizador deve enviar ao módulo de simulação todos os dados informados pelo usuário na interface de parametrização do modelo, e transformar esses dados em uma fonte que será usada pelo interpretador a executar o modelo. Além disso, o módulo de simulação deve permitir a definição do tempo de simulação. Neste contexto, podemos ver o módulo de simulação como um *parser* que gera código-fonte em diversas linguagens.

O módulo de análises não existe na ferramenta atual, somente um *mockup* de como é sua interface foi criada está presente. Esse módulo deve coletar todos os resultados produzidos ao fim da simulação e permitir que o usuário carregue resultados prévios da execução do modelo a fim de realizar comparações matemáticas ou visuais. Uma função adicional e interessante para DengueME neste módulo seria a publicação de resultados. QT permite a integração com Cloud e um repositório de resultados para diferentes modelos seria útil para que outros pesquisadores estudassem estes resultados e os utilizassem como forma de comparação. Além disso, cada modelo poderia ter seus exemplos de resultados com os seus respectivos parâmetros salvos em nuvem, se o usuário desejar que os mesmos sejam carregados. O módulo de análises é complexo

para ser implementado e depende do módulo de simulação para funcionar. O calibrador é outro módulo dependente de outros, já que é necessário que o módulo de análises ou que o módulo de simulações estejam implementados e funcionais. Atualmente existe um *mockup* de sua tela e classes básicas para seu controle, ou seja, está em construção. O calibrador deve contar com alguns algoritmos de calibração, que devem ser separados em diferentes classes para facilitar a manutenção e inserção de novos algoritmos. Cada algoritmo ou técnica de calibração ofertada tem seus próprios impactos e forma de trabalhar sobre os componentes do modelo. Esse é um módulo complexo e que precisa de um estudo mais especializado para sua implementação. O editor de modelos existe dentro do DengueME, mas carece ainda de algumas funcionalidades básicas como: opção de carregamento de múltiplos arquivos fonte, lógica para versionamento dos fontes dos modelos e opção de executar o fonte para testes. Ao modificar o fonte de um modelo por conta de alguma análise feita no calibrador ou algum novo *insight* vindo dos resultados, o usuário deve ser capaz de salvar esse novo modelo como uma nova versão do antigo, sem interferir no já existente. O pacote de modelo seria copiado e somente o fonte modificado seria substituído, e caso o usuário não altere nenhuma variável de entrada, a interface de parametrização se mantém inalterada. A ferramenta deve informar ao usuário se alguma parte da interface de parametrização precisa ser atualizada.

Por fim, o editor de dados não existe em DengueME, mas é um dos pontos-chave do *framework*. Interagir com dados desconhecidos é uma dificuldade para qualquer pessoa, já que cada modelo tem seu próprio conjunto de entradas e espera-se que estejam formatadas da maneira correta, do contrário o modelo não funcionará como esperado. O editor de dados é composto de duas partes: 1) O construtor de editor de dados, que está ligado ao construtor de pacotes de modelos, que é onde o usuário monta o editor com todas operações necessárias para eventuais tratamentos da entrada de seu modelo; 2) O editor de dados propriamente dito, que o usuário utiliza para auxiliar na parametrização do modelo. Ambos precisam ser implementados em DengueME, mas para isso ainda é necessário um estudo sobre quais operações seriam úteis para os modelos de dengue da atualidade. Por exemplo: diversos modelos fazem uso de dados de temperatura, e portanto o editor de dados poderia contar com ferramentas para formatar esses dados no padrão esperado pelo modelo.

4.4 Documentação

A documentação original do DengueME contava com alguns tutoriais sobre como fazer o uso básico da ferramenta, mas não ensinava sobre as ferramentas para modeladores. Um dos principais problemas da documentação é que ela não leva em conta a possibilidade da expansão do software; logo, ela não explica ou demonstra como seria possível fazer isto.

Para expandir o software é preciso entender a arquitetura do código e o que se tem funcional e não funcional. É necessário apresentar o *framework* em que DengueME foi baseado para que outros desenvolvedores tenham uma noção geral da arquitetura do sistema e de suas partes. Documentar quais são as ligações existentes entre os módulos é necessário para se manter o registro do que está pronto ou não no *framework*. Além disso, deve ser demonstrado na documentação quais módulos faltam e do quê cada qual ainda necessita.

Outra necessidade da documentação do DengueME que é fonte grande de dúvidas para modeladores é: Como deve ser o código de meu modelo para que o mesmo funcione em DengueME? Por mais que o *framework* estabeleça a flexibilidade de linguagens, para o bom funcionamento do *parser* é necessário um padrão para certas partes do código. Esse padrão ajuda o *parser* a gerar um arquivo de entrada correto para o modelo.

A exemplo de TerraME, para que um modelo seja inserido em DengueME, o fonte do modelo deve ter todas suas variáveis de entradas declaradas como locais e sendo atribuídas a ela uma variável global inexistente de mesmo nome. Ao criar a interface de parametrização, o usuário insere nos campos os nomes de cada uma das variáveis. O usuário então parametriza cada uma delas no parametrizador e envia para o módulo de simulação, que por sua vez passa para o *parser*. O *parser* recebe as informações dos nomes das variáveis e seus valores e gera um arquivo na linguagem TerraML contendo todas as variáveis definidas como globais e seus respectivos valores. Ao final desse arquivo existe uma chamada para o arquivo main do código-fonte do modelo que esta dentro do pacote de modelos.

O processo acima, atualmente, não é explicado em nenhuma parte do DengueME, desencorajando novos modeladores a colocarem seus modelos na ferramenta, uma vez que eles não tem ideia de como padronizar seu código. Neste contexto, junto da descrição da arquitetura nova da ferramenta, documentos de como padronizar códigos das linguagens atualmente suportadas por DengueME foram escritos. Além disso, foi elaborado um guia sobre como adicionar ao fonte do DengueME novas plataformas de modelagem.

4.5 Considerações

Todas alterações em DengueME estão disponíveis online em um repositório do github - https://github.com/saraiva3/DengueME_Refactor. O projeto está sob a licença MIT 3, que permite a distribuição e modificação livre de todas as partes que o compõem, permitindo assim que qualquer usuário interessado use o código do DengueME como referência ou ponto de partida para sua própria ferramenta de modelagem e simulação.

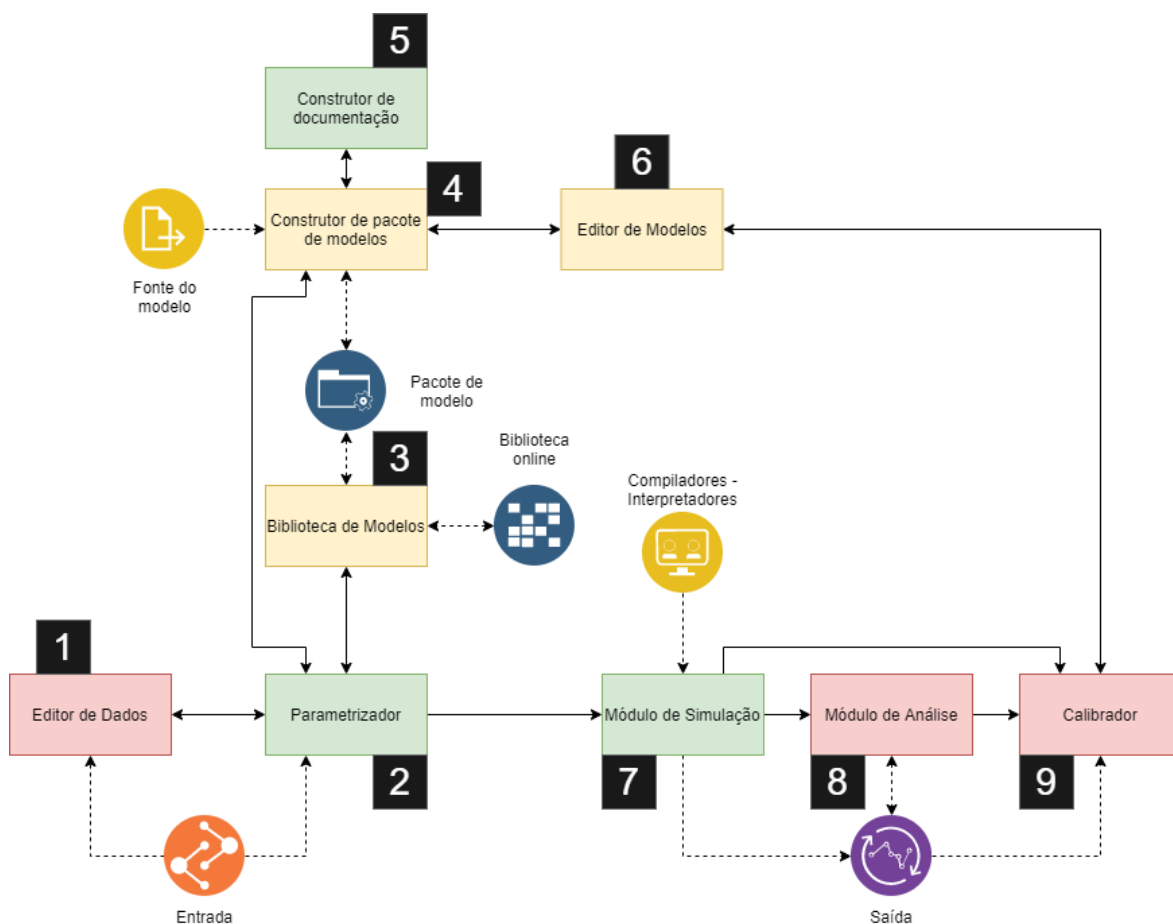


Figura 4.2. Estado atual do DengueME.

A figura 4.2 representa o atual estado do DengueME. Os módulos em verde estão funcionais e aptos ao uso. Os em amarelo estão em estado de desenvolvimento ou carecem de ajustes que impedem seu funcionamento total (por exemplo, o editor de modelos carece ainda de um sistema para salvar as modificações), e os em vermelho estão em planejamento ou fase de prototipagem.

Muito precisa ser feito em DengueME para que ele alcance as expectativas que o *framework* traz. Tendo todos os módulos implementados e a ferramenta sendo testada

por usuários reais, seria possível conduzir uma reavaliação do *framework* e verificar partes em falta ou que possam ser melhoradas.

Capítulo 5

Estudos de Caso

Neste capítulo são apresentados dois estudos de caso para o *framework*, utilizando a ferramenta DengueME refatorada e adaptada conforme descrito no capítulo anterior. As partes não implementadas são indicadas por meio de protótipos, acompanhadas de explicações sobre como utilizar.

No caso de uso 1, um modelo em TerraML, representando um modelador que deve passar por todo o processo de criação do pacote de modelos, parametrização do modelo, simulação e visualização dos resultados. No caso de uso 2 é apresentado um *toy model* em R, representando o modelador que deseja somente parametrizar e executar um modelo pronto de algum fenômeno. Os casos de uso visam demonstrar todas partes do *framework* em ação, preenchendo os *gaps* apresentados na revisão bibliográfica e demonstrando como o usuário é auxiliado durante o processo.

Uma vez que a ferramenta DengueME não conta com todos os módulos e *features* do *framework*, as partes inexistentes são apresentadas por meio de mockups e explicações de como elas podem ser implementadas e como funcionariam.

5.1 Caso de Uso 1

O primeiro caso de uso demonstra como um usuário modelador utilizaria uma ferramenta baseada no *framework*. Assumimos aqui que o usuário tem conhecimento em programação e que o código inicial do modelo já está pronto. O modelo utilizado aqui é um modelo SEIR (Suscetível, Exposto, Infectado, Recuperado) [Kermack & McKendrick, 1991] baseado em agentes para a transmissão da dengue. O modelo de agentes assume que cada indivíduo da população é um agente. Estes agentes se movimentam em uma grade celular de tamanho variável, e a movimentação dos agentes é ditada pelas regras de vizinhança que a grade suporta. Em alguns pontos da grade existem

focos de dengue, e ao passar por ali existe uma chance do agente contrair a doença. O modelo foi implementado na linguagem TerraML para o interpretador TerraME.

Pela descrição do modelo podemos identificar as entradas necessárias para executar o modelo: taxas de transmissão, exposição e recuperação, tempo de simulação, número de passos para cada agente, população total de agentes, tamanho da grade, regra de vizinhança e posicionamento inicial de cada agente. Dentre essas variáveis, duas podem vir de dados geográficos: a primeira seria a população, onde o usuário poderia ter dados da população de bairros ou cidades e utilizar os mesmos em uma grade de tamanho equivalente ao local atualizado onde cada indivíduo da população seria um agente; A segunda é a grade em si, que poderia ser construída a partir da geometria do local de estudo. Assumimos que a população vem de dados geográficos e a grade é gerada pela ferramenta TerraME.

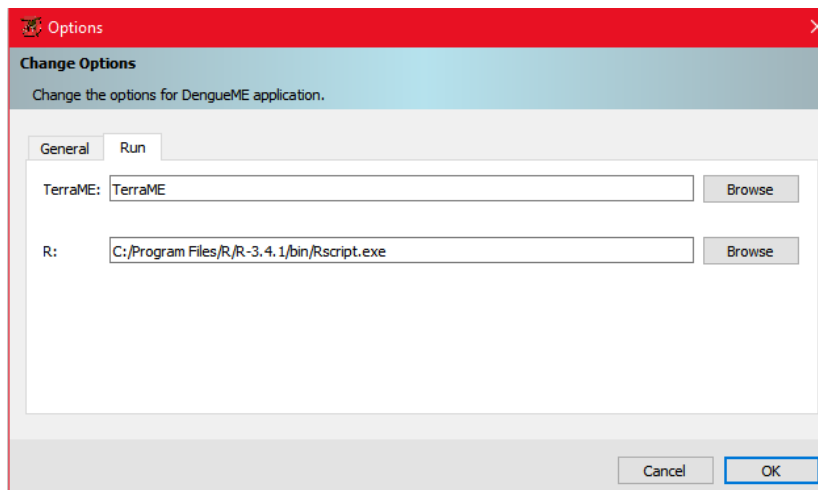


Figura 5.1. Configurações DengueME.

O primeiro passo para uso da ferramenta consiste em abrir a mesma e ir até as configurações (Figura 5.1), para verificar na aba *run* se a ferramenta já está com o TerraME e outros interpretadores configurados. Feito isso, o usuário deve ir ao construtor de pacotes de modelos localizado no menu **Tools => Construtor de Pacote de Modelos**. Aberto o construtor (figura 5.2), o usuário pode agora criar um projeto para seu modelo e iniciar a configuração das partes que compõem o pacote de modelos: Interface de parametrização, Documentação do Modelo, Carregamento do Fonte do Modelo e Parâmetros Editor de dados.

A primeira tela do construtor de pacote de modelos, apresentada na Figura 5.3, é a tela de inserção dos dados básicos do modelo, informações dos autores, carregamento dos arquivos fonte do modelo e configuração do interpretador que o modelo utiliza

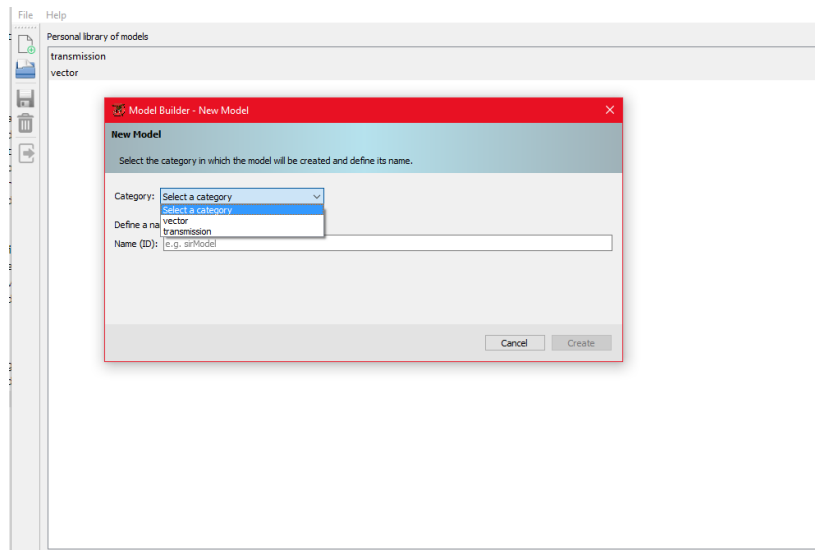


Figura 5.2. Tela inicial construtor de pacote de modelos DengueME.

(como demonstrado na 5.3). É importante que seja indicado também qual dos arquivos-fonte contém a função *main* do modelo, para que o mesmo seja executado na hora da simulação. Seguindo o fluxo da ferramenta, o usuário é guiado a criar a interface de parametrização de seu modelo.

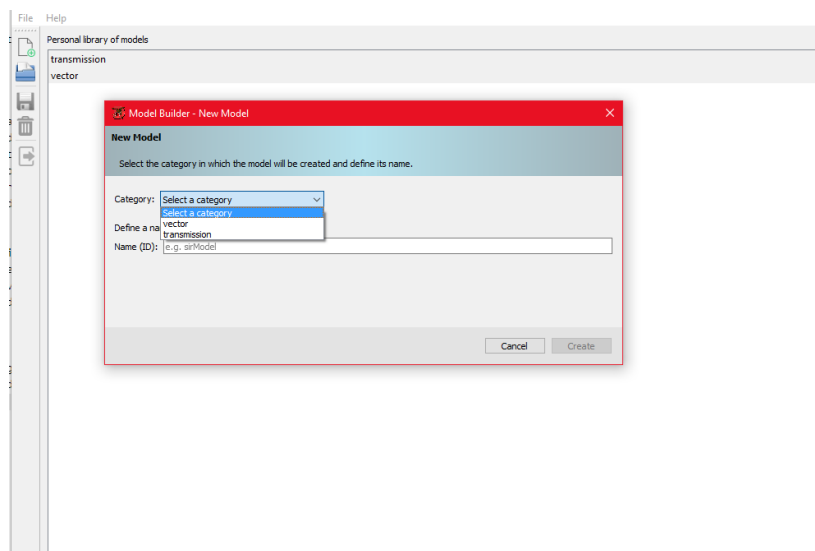


Figura 5.3. Informações básicas do modelo.

A interface de parametrização é um dos pontos chave do *framework*, é através dela que todas entradas serão inseridas e por isso é necessário que seja feito um mapeamento entre entrada e *widgets* de tela. A ferramenta deve contar com diferentes *widgets* para a montagem da interface. Para este modelo, a interface da entrada utiliza campos para

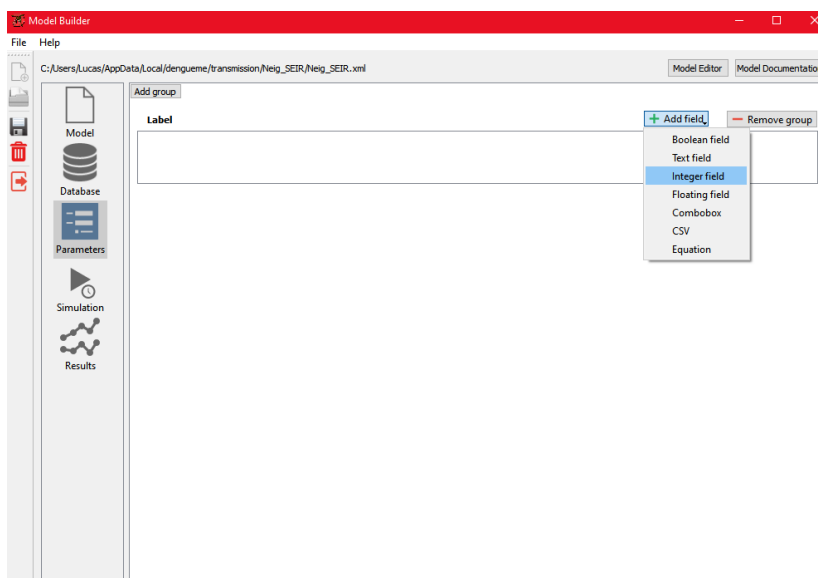


Figura 5.4. Criação da interface de parametrização.

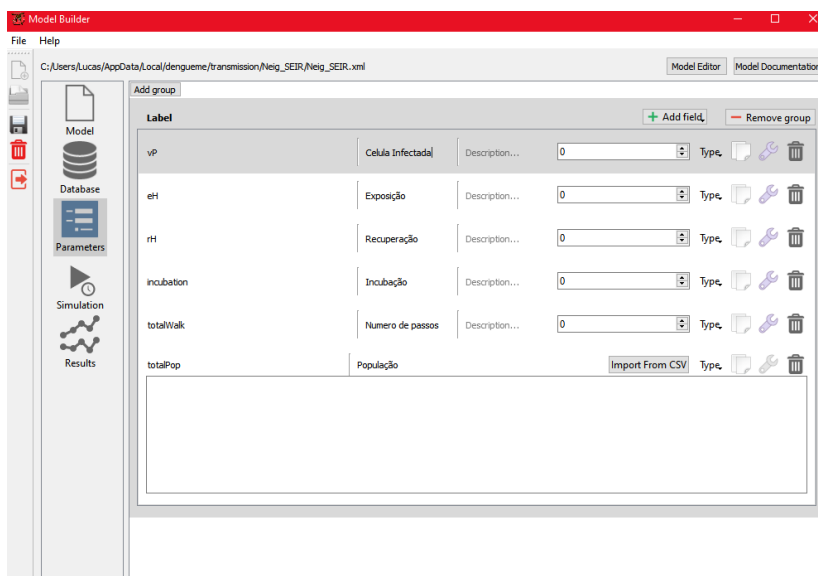


Figura 5.5. Interface de parametrização pronta.

inserção de valores numéricos e um *widget* para carregar uma planilha com dados de população. A figura 5.4 apresenta o construtor de interfaces do DengueME enquanto que a figura 5.5 apresenta a interface de parametrização para o modelo deste caso de uso.

A saída do modelo é composta de dois gráficos e um mapa. Atualmente DengueME não contém *widgets* para mapas, logo não será possível editar as variáveis de configuração do mapa, mas é possível alterar as configurações dos dois gráficos e esco-

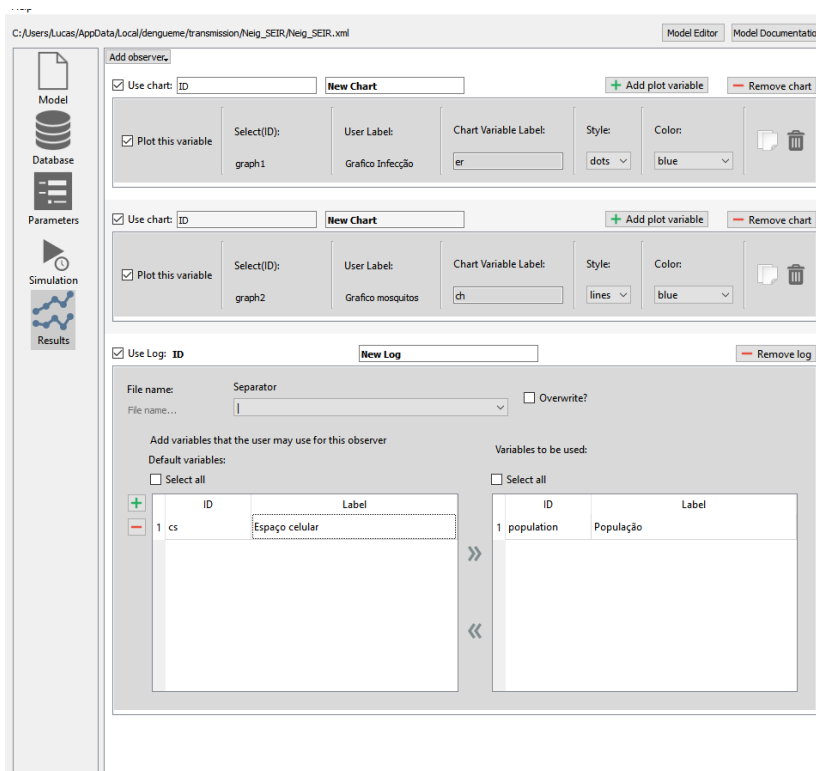


Figura 5.6. Interface de saída pronta.

lher as opções do arquivo CSV a ser gerado contendo resultados. Como informado, os *widgets* de saída correspondentes a estas apresentações no TerraME já estão configurados no construtor de interfaces, bastando somente o usuário escolher eles e definir o nome dos campos de configuração da saída com base no código-fonte como demonstrado na figura 5.6.

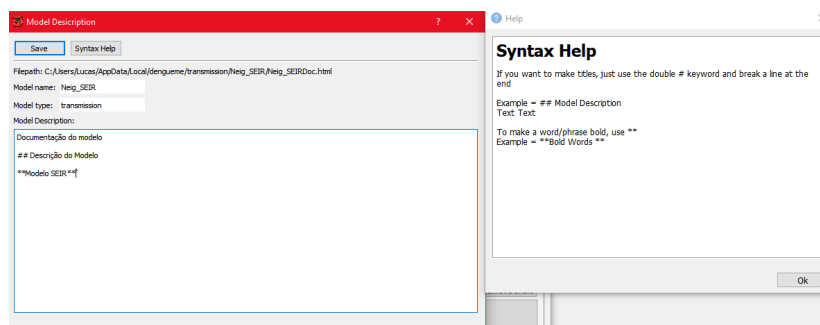


Figura 5.7. Interface de saída pronta.

Montada a interface, o usuário pode agora construir a documentação de seu modelo e configurar os parâmetros do editor de dados. O editor de documentação é acessado no DengueME pelo botão "*Editor de Documentação*", o mesmo é apresentado

na figura 5.7. Um editor de texto interno a ferramenta é aberto para que o usuário crie a documentação do modelo. O editor deve contar com uma *markup syntax* embutida nele (preferencialmente baseada em html) para melhor estruturar o texto e deixar a apresentação mais chamativa. A documentação pode ser salva em PDF e HTML, permitindo a visualização online e local.

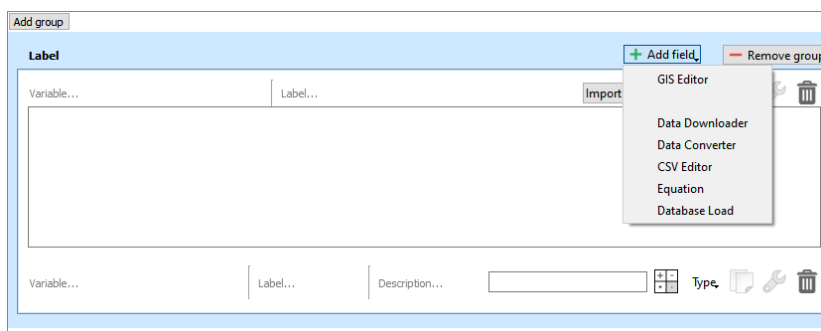


Figura 5.8. Prototipo construtor do editor de dados.

Criada a documentação, o usuário deve agora configurar os parâmetros e interface do editor de dados. O construtor do editor de dados não está implementado em DengueME, mas um modelo de seu funcionamento é apresentado na Figura 5.8. O mesmo é identificado como uma parte importante do *framework* e do processo de criação do pacote de modelos, uma vez que usuários que vierem a utilizar o modelo possam ter dificuldades em tratar seus dados de entrada para um modelo desconhecido a eles. Baseado nos *widgets* de tela escolhidos na montagem da interface de parametrização, o editor de dados deve ter noção dos tipos de operações que podem ser realizadas sobre o dados a fim de auxiliar o usuário. Por exemplo: supondo que o usuário tenha o dado de população em um formato csv distribuído em 20 anos, e em sua interface de parametrização inseriu um *widget* para carregar arquivos csv, o editor de dados pode oferecer uma operação que consiga extrair células ou linhas específicas do arquivo para o usuário utilizar como *input*, outra operação possível seria fazer uma medida da população no arquivo csv ou até mesmo criar uma série temporal para que simulações em diferentes anos e populações sejam realizadas no batch de simulações. Um segundo uso seria a oferta de operações para dados geográfico. Supondo que a grade do modelo fosse carregada por meio de um arquivo *shapefile*, o editor de dados poderia mostrar as propriedades desse *shape* e se indicar se ele necessita de alguma transformação, como recorte ou transformação de coordenadas.

Finalizado o pacote de modelos, o usuário modelador pode agora inserir seu pacote na biblioteca online de modelos ou mantê-lo somente em sua máquina. Para inserir o pacote de modelos na biblioteca basta acessar a biblioteca e escolher a opção de Upload,

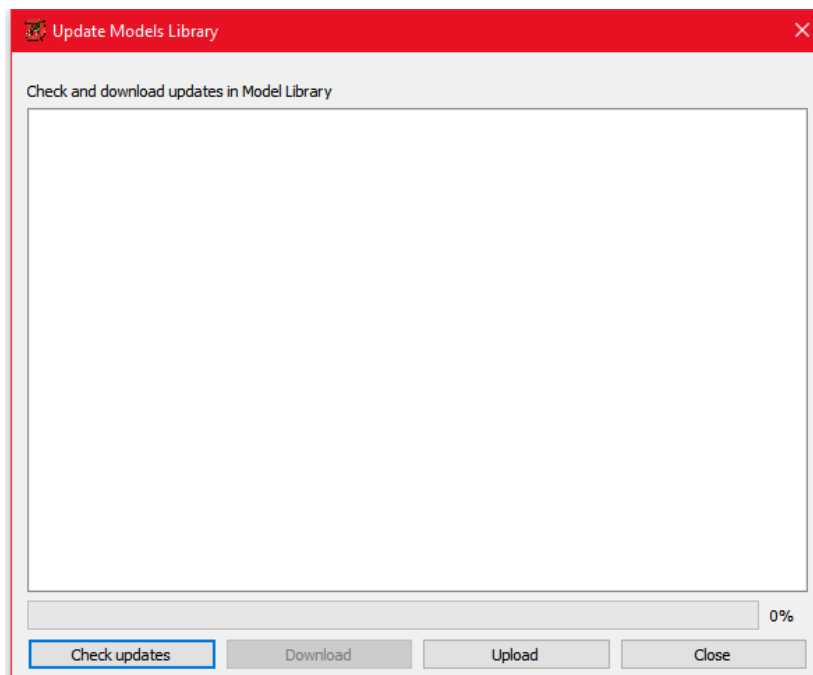


Figura 5.9. Biblioteca de modelos.

figura 5.9. O modelo será disponibilizado online juntamente de sua documentação, interface de parametrização e interface do editor de dados.

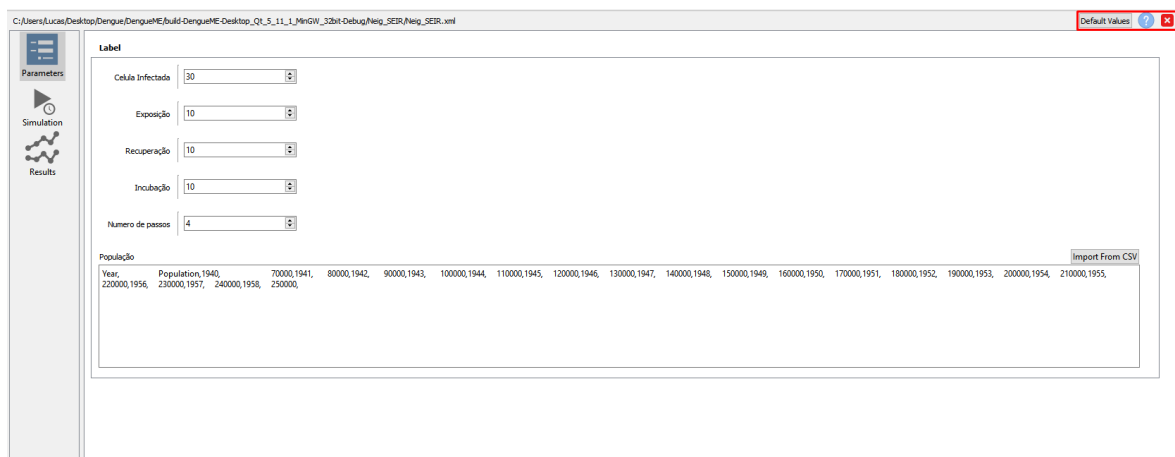


Figura 5.10. *Dashboard* com interface de parametrização do modelo.

Feito isto, é possível criar um novo projeto contendo o modelo para ser parametrizado e simulado. Ao criar um projeto com seu modelo, a interface de parametrização será apresentada na *dashboard*, figura 5.10 para que os dados de entrada sejam inseridos no modelo. A interface de parametrização da saída também é apresentada aqui em uma aba diferente, ambas devem ser preenchidas para que o modelo funcione como

esperado.

Se necessário, o usuário pode abrir o editor de dados pelo menu "*Tools -> Editor de dados*". No editor de dados o usuário pode checar quais dados são necessários para a entrada do modelo e se, por algum motivo, existem tratamentos que podem ser feitos sobre os dados. Por exemplo, se o modelo aceitasse um arquivo *shapefile* para o mapa de saída, o mesmo poderia ser recortado ou transformado aqui, dependendo do formato no qual se encontra. Para o modelo aqui apresentado não existe a necessidade do uso do editor de dados, mesmo que tenhamos configurado um. A ideia central do editor de dados é oferecer suporte para o tratamento de dados de entrada e ajudar no entendimento de quais dados são necessários ao modelo. As operações que ele pode realizar ficam limitadas a o que for implementado utilizando a tecnologia escolhida para criação do *framework*. Por exemplo, em QT, seria possível oferecer diversas operações sobre dados geográficos, pois QT tem acesso à API do QGIS e à biblioteca GRASS para manipulação de dados geográficos. Um exemplo do editor de dados aberto está na figura 5.11

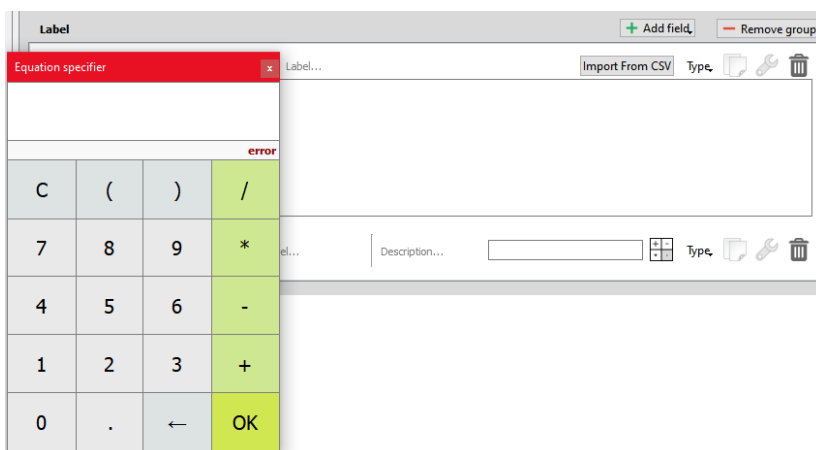


Figura 5.11. Mockup editor de dados.

Para maior entendimento do modelo, a documentação ligada ao mesmo pode ser acessado pela *dashboard* (a mesma que se encontra o parametrizador). A documentação lista em detalhes as nuances do modelo e do objeto de estudo. Aqui também deve ser descrito um conjunto exemplo de dados onde se sabe o resultado para que o usuário os utilize de teste e se familiarize com a ferramenta. Para este modelo não foi construída uma documentação, uma vez que o foco aqui são as etapas e processos do *framework* e não o modelo em si.

Tendo tudo parametrizado, o usuário deve clicar em *Simular* para abrir o módulo de simulação. Neste módulo é possível informar o tempo de simulação, em nosso caso

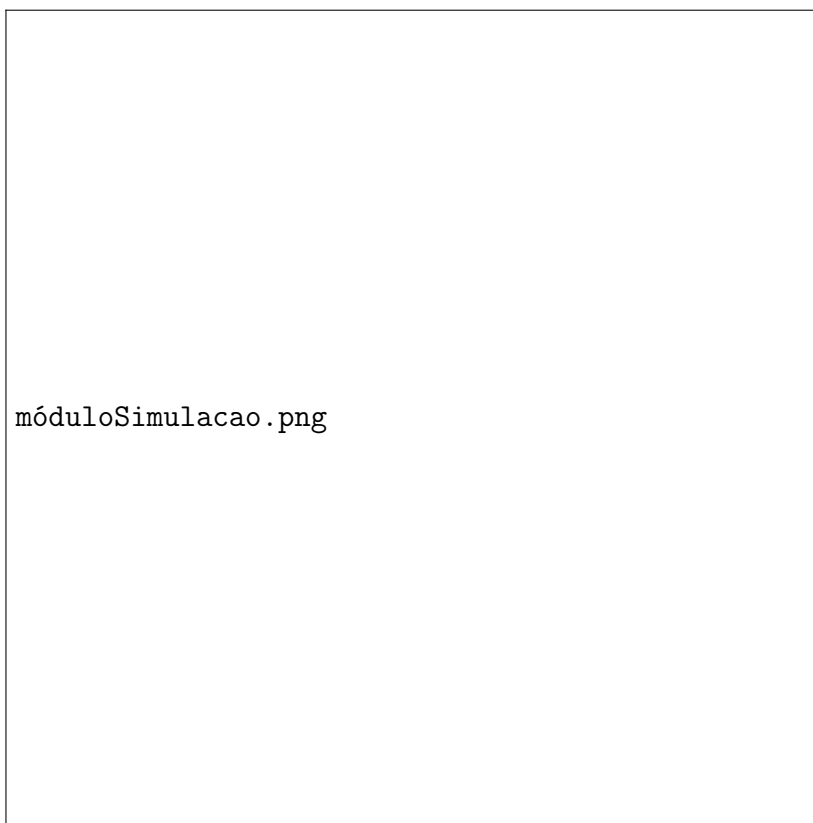


Figura 5.12. Mockup módulo de simulação.

os ticks de simulação e realizar a criação de um batch de simulações. Neste exemplo não iremos criar um batch de simulação pois essa funcionalidade não está implementada em DengueME e seria difícil simular seu funcionamento por meio de mockups. A ideia central é ser possível definir quantas vezes deseja executar o mesmo modelo (ou algum outro da biblioteca), se os parâmetros de entrada se mantêm - no caso de modelo igual, e caso não, parametrizar as outras instâncias de seu modelo. A figura 5.12 apresenta o Mockup de como seria um módulo de simulação. Com tudo configurado podemos agora executar o modelo.

Ao clicar em executar, a ferramenta irá gerar o arquivo fonte contendo os dados de entrada e saída inseridos no parametrizador e iniciar a execução do modelo. Ao final da simulação as saídas do modelo são mostradas na tela, figura 5.13. Para este modelo alguns dados são salvos em disco - os valores dos gráficos são salvos em um arquivo CSV.

O analisador é representado por um mockup na figura 5.14, e o calibrador é demonstrado na figura 5.15. Ambas são módulos do *framework* com uma interface bem similar e servem para ajudar no entendimento do modelo, do objeto de estudo e para a busca de possíveis problemas no modelo. O foco deste trabalho não é a análise

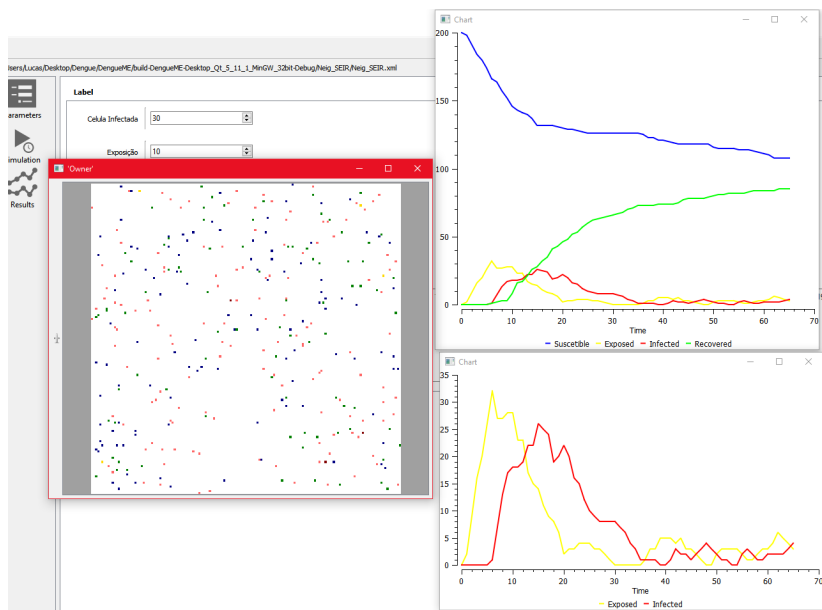


Figura 5.13. Resultados do modelo.

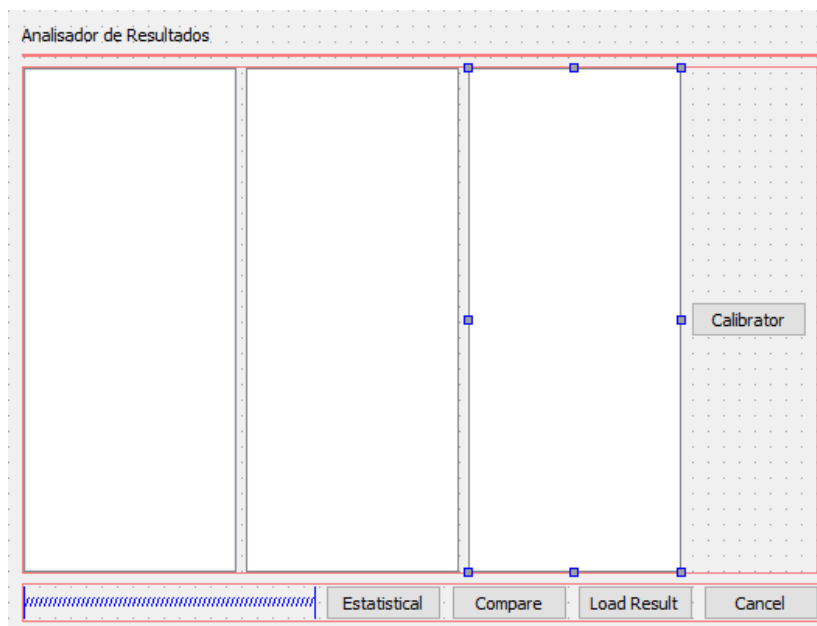


Figura 5.14. Mockup analisador de resultados.

do modelo aqui apresentado, logo, não foi realizada uma calibragem do modelo nem uma análise profunda dos resultados. O importante aqui é demonstrar que o usuário de dentro da ferramenta deve ter acesso a *features* que permitam o mesmo analisar, avaliar e validar seus modelos e suas saídas. Métricas podem ser implementadas dentro do analisador de dados para que o usuário saiba como se guiar daqui pra frente, se deve ou não alterar seu modelo e se pode realizar tomadas de decisões pelos resultados deste

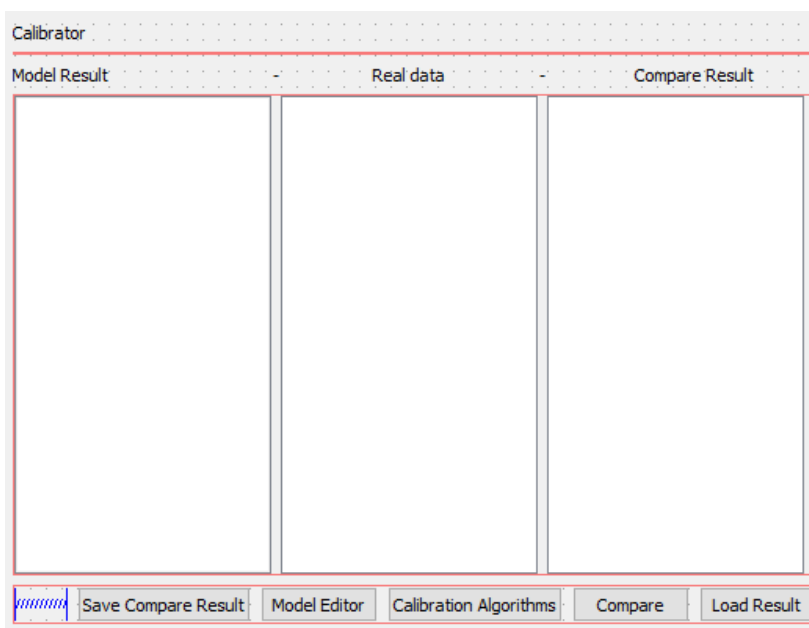
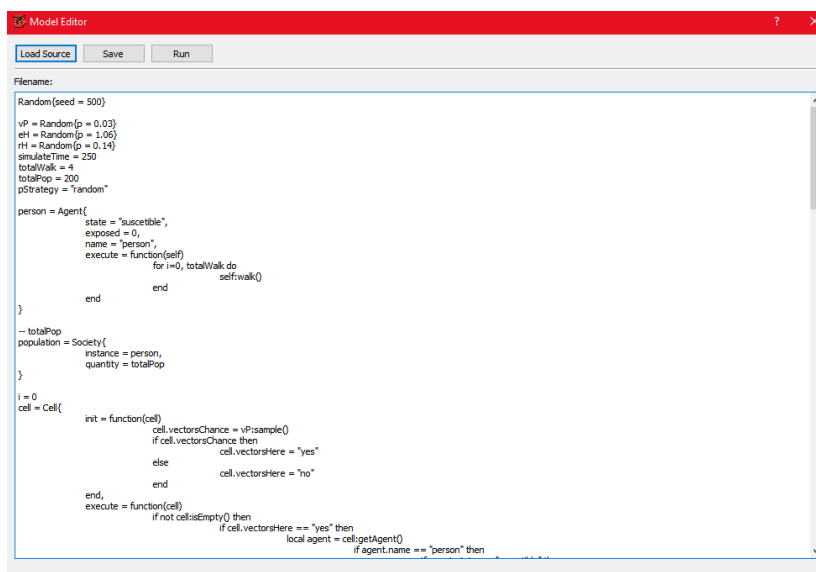


Figura 5.15. Mockup calibrador de modelo.

modelo. Tais métricas seriam dependentes de que tipos de modelos a ferramenta se propõe a cobrir, logo, não é definido aqui um conjunto padrão de métricas associadas ao analisador de resultados.

Para calibrar o modelo, é necessário que o usuário tenha dados reais relacionados ao fenômeno e local de estudo, para que assim ele possa comparar os resultados de seu modelo com estes e verificar o quão próximas as fórmulas e regras implementadas no modelo chegam dos dados reais. Além disso, o calibrador pode ter técnicas de calibração implementadas para que se utilize e avalie a confiabilidade e validade do modelo. Futuramente, pode-se investigar técnicas de calibração genéricas que poderiam ser implementadas em ferramentas que venham a se basear no *framework* proposto.

Para este caso de uso assumiremos que existem alguns problemas nas regras do modelo e iremos alterar elas diretamente de dentro da ferramenta. O editor de modelos deve ser aberto, figura 5.16, o mesmo irá mostrar o fonte do modelo desejado. O fonte pode ser modificado sem que a interface de parametrização ou outras partes do pacote de modelo sejam afetadas. A única forma de afetar a interface de parametrização seria alterando o tipo ou nome de alguma das variáveis de entrada e saída do modelo. Feitas todas as alterações, o usuário pode optar entre substituir a primeira versão do modelo ou salvar uma nova versão. Caso opte pela segunda opção, se nada na entrada e saída foi modificado, uma cópia de todo o pacote de modelos será feita e o usuário contará com duas versões do modelo. Caso as entradas sejam modificadas, o usuário é alertado e deve voltar ao construtor de pacote de modelos para ajustar as partes modificadas.



```

Model Editor
-----
Load Source Save Run
Filename:
Random(seed = 500)
vP = Random(p = 0.03)
eH = Random(p = 1.06)
rt = Random(p = 0.14)
simulateTime = 250
totalWalk = 4
totalPop = 200
pStrategy = "Random"

person = Agent(
  state = "susceptible",
  exposed = 0,
  name = "person",
  execute = function(self)
    for i=0, totalWalk do
      self.walk()
    end
  end
)

-- totalPop
population = Society(
  instance = person,
  quantity = totalPop
)

i = 0
cell = Cell(
  init = function(cell)
    cell.vectorsChance = vP:sample()
    if cell.vectorsChance then
      cell.vectorsHere = "yes"
    else
      cell.vectorsHere = "no"
    end
  end,
  execute = function(cell)
    if not cell.isEmpty() then
      if cell.vectorsHere == "yes" then
        local agent = cell:getAgent()
        if agent.name == "person" then

```

Figura 5.16. Editor de modelo.

Agora o usuário tem duas versões de seu modelo e pode parametrizar ambas através do *batch* de simulações e comparar resultados no analisador, repetindo assim todo o ciclo de modelagem. Dessa forma a evolução do modelo ocorre de forma mais natural, com a ferramenta oferecendo todo suporte na parte computacional e o usuário tendo somente o trabalho de operar a ferramenta e entender do objeto de estudo. O *framework* visa oferecer o máximo de suporte na medida do possível, de uma forma amigável e que possa ser factível.

5.2 Caso de Uso 2

O caso de uso 2 é mais curto que o caso de uso 1 pois visa analisar o cenário de um usuário sem conhecimentos em programação, mas que tem interesse em parametrizar e simular um modelo que já exista na biblioteca de modelos da ferramenta. Essa tarefa é simples e rápida, e aqui demonstramos como a base do *framework* consegue suportar ferramentas que sejam úteis para tais tarefas.

Ao abrir a ferramenta, a primeira coisa a se fazer é ir até a biblioteca de modelos e checar se ela está sincronizada com a biblioteca online. É importante que a biblioteca de modelos esteja sempre atualizada e com modelos pertinentes aos objetos de estudo que a ferramenta representa. Por exemplo, a biblioteca de modelos do DengueME conta com todos modelos de transmissão de dengue clássicos (SIR, SEIR, SIRSI) e um modelo básico de ecologia do vetor.

Sincronizada a biblioteca, o usuário deve escolher qual modelo vai parametrizar e

se atentar a qual plataforma de modelagem ou interpretador/compilador aquele modelo utiliza. A ferramenta deve sempre avisar o usuário caso a mesma não esteja configurada e oferecer meios de baixar/instalar o que for necessário de maneira rápida e fácil, auxiliando o usuário na configuração do ambiente. Para este caso escolhemos um modelo SIR em R.

Uma das características do *framework* é a flexibilidade de usar interpretadores/compiladores variados para diferentes modelos. Aqui podemos ver a possibilidade da existência desta flexibilidade e ela em funcionamento. A ferramenta DengueME já suporta TerraME e R para seus modelos, oferecendo ao usuário mais opções do que normalmente outras ferramentas ofertariam.

O usuário pode agora criar um projeto contendo o modelo que deseja parametrizar e simular. O modelo aparece na árvore de navegação, e ao clicar nele a interface montada pelo modelador que inseriu este modelo na biblioteca é carregada. Partindo deste ponto, a melhor forma de entender o modelo e suas entradas é abrir a documentação clicando no botão de help para ler um pouco sobre suas regras. Outra forma de ajudar no entendimento do modelo e suas entradas é abrir o editor de dados que foi previamente montado.

De dentro do editor de dados o usuário terá uma noção de tudo que ele irá precisar para parametrizar o modelo e se seus dados precisam de algum tratamento. No caso do modelo aqui apresentado, nenhum dado precisa de tratamento pois o usuário já tem todas taxas anotadas em suas notas de estudo. O editor de dados pode ser fechado e a documentação checada. A documentação explica como foi montada a regra do modelo e quais seus detalhes internos. Além disso, a própria documentação conta com apresentação de valores *default* e qual deveria ser seu resultado, fornecendo o usuário uma forma de testar se seu modelo está normal.

Este é um modelo simples de ser parametrizado, contando com algumas entradas numéricas (taxas) e uma saída em CSV que guarda os valores das variáveis desejadas. A parametrização da saída é feita aqui também em outra aba. Este é um tipo de modelo que se faz interessante ter um batch de simulações. Tendo o modelo somente taxas como entradas, testar várias taxas de diferentes estudos é interessante para o usuário entender o impacto de cada uma delas no modelo e como aplicar isto em seus estudos. Como as saídas são salvas em CSV, o analisador de resultados conseguiria ler estas e apresentar os dados de forma que o usuário consiga facilmente analisar eles, por exemplo, organizar em uma tabela ou traduzir os dados em um gráfico.

Parametrizado o modelo, o usuário pode agora abrir o módulo de simulação para configurar o tempo de simulação, neste caso, a quantidade de vezes que a regra será executada. É possível checar aqui se estamos usando a versão correta do interpretador

necessário e qual o caminho configurada para eventuais saídas do modelo. Ao clicar em iniciar, uma chamada de sistema é feita para interpretador da linguagem R, passando como parâmetro o código-fonte com as configurações de entrada e saída gerados pelo módulo de simulação. O modelo é executado e suas saídas são salvas em disco em arquivos csv.

Para este modelo faremos mais 4 execuções alterando as taxas de transmissão e de recuperação, para facilitar iremos utilizar o batch de modelos dentro do módulo de simulação. O batch de modelos pede qual modelo será executado e quantas vezes ele deve ser simulado 5.12. Ao clicar no botão "Parametrizar" a interface de parametrização do modelo é aberta. O usuário tem a opção de parametrizar uma delas e copiar os mesmos parâmetros para todas para que seja modificado em cada uma delas somente o parâmetro desejando, acelerando o trabalho de parametrização de várias instâncias e mantendo a consistência das entradas inalteradas entre as instâncias.

No caso do DengueME, o batch de modelos não existe. Seria possível implementar o mesmo de forma que todos modelos executassem em paralelo, uma vez que QT oferece vasto suporte a multi-threading e é possível abrir múltiplos interpretadores de R ao mesmo tempo. Executada todas instâncias o analisador abre automaticamente, com todas as saídas já carregadas em sua interface. As saídas são agrupadas com sua devida instância em "pastas". Ao selecionar duas ou mais pastas e quais saídas deseja comparar as mesmas são mostradas nas tela auxiliares 5.13.

Para este caso, assumimos que o modelo esta calibrado, pois o mesmo já se encontra disponível na biblioteca de modelos. Ainda assim, sabemos que a calibragem pode ser diferente para um conjunto de dados especifico, é interessante abrir o calibrador para se comparar os resultados obtidos com dados reais de seu caso de estudo. A checagem ajuda o usuário a entender se aquele modelo realmente atende suas necessidades e se consegue representar bem a situação de seu objeto de estudo. Nada impede o usuário de realizar modificações no modelo e gerar sua própria versão, mas para tal é necessário conhecimento na linguagem de programação usada para construção do mesmo. Uma forma de resolver isso é o usuário utilizar a documentação para checar o contato do autor do modelo (caso disponível) e conversar com o mesmo para uma parceira de estudo ou dicas de como modificar o modelo. Assim, através da ferramenta se cria uma comunidade de estudos que pode trabalhar junta melhorando e criando modelos.

Podemos ver aqui que o processo de parametrização é bem simples e intuitivo ao usuário, o oferecendo um passo a passo de como parametrizar o modelo e indicando tudo necessário para funcionamento do tal. Em outras ferramentas de modelagem aqui apresentadas, o usuário tinha acesso somente a uma interface de parametrização,

algumas vezes sem muita explicação de como ela funciona e do que representa cada parâmetro. Quanto mais informações o modelo conter, seja na sua interface de parametrização, editor de dados ou documentação, isso conta como auxílio ao processo de modelagem e simulação.

5.3 Comparativo

Ambos os casos de uso tentam passar pelos diversos módulos do *framework*, utilizando a ferramenta DengueME, por diferentes perspectivas com o intuito de demonstrar todas as possibilidades que o mesmo oferece. O primeiro caso demonstra a aplicação e uso do *framework* para um usuário que tenha programado um modelo para um dado fenômeno e que deseje utilizar a ferramenta para parametrizar, avaliar, calibrar e compartilhar seu modelo. Aqui, o *framework* não serve somente para inserção de dados no modelo mas sim para cobrir diversas etapas do ciclo de modelagem, auxiliando o usuário a melhorar seu modelo para que o mesmo seja compartilhado em uma biblioteca online, que conseqüentemente cria uma comunidade em torno dela.

O caso de uso número dois demonstra o usuário que deseja utilizar algum modelo disponível na biblioteca de modelos da ferramenta. O mesmo tem dados para alimentar o modelo, ou descobrira quais dados precisa coletar, e quer utilizar o modelo para seus estudos ou trabalhos de exploração. O usuário terá acesso a ferramenta para comparação e análise dos resultados obtidos pela simulação, comparação com dados reais que o mesmo tenha acesso e possibilidade de parametrizar de quantas formas desejar o mesmo modelo, mantendo todos resultados salvos para futuras observações.

As outras ferramentas apresentadas na revisão bibliográfica não contam com todas as *features* do *framework*, como demonstrado na tabela 2.1. As seguintes atividades não estão disponíveis em outras ferramentas, e o *framework* proposto visa cobri-las:

1. O editor de dados que permite a manipulação dos dados que o modelador tem para a entrada de seu modelo podem ser feitos de dentro da ferramenta
2. Não seria possível calibrar o modelo de dentro da ferramenta com base nos resultados reais e os resultados obtidos pela simulação. .
3. Os resultados reais não poderiam ser comparados de dentro da própria ferramenta.
4. A modificação do fonte do modelo pode ser realizada no *NetLogo*, mas o mesmo não conta com um mecanismo automático de versionamento de modelos, que é idealizado aqui no *framework* como parte de suas *features*.

5. A ferramenta permite o uso de modelos de diferentes interpretadores/compiladores, dando flexibilidade ao usuário e ao desenvolvedor dos modelos.
6. Possibilidade de execução em paralelo ou fila de N instâncias de modelos para comparativo dos resultados.

O *framework* é uma forma de guiar desenvolvedores a criar ferramentas de parametrização que ofereçam além da interface de parametrização e demonstração de resultados. Contando com elementos de edição de código e interface, interação com bibliotecas de modelos online, calibradores e analisadores de resultados, o *framework* consegue cobrir diversas tarefas que normalmente se é necessário usar diversas ferramentas ao mesmo tempo.

Para uma avaliação mais robusta do *framework* em termos de usabilidade, seria necessário comparar as ferramentas aqui apresentadas com uma inteiramente baseada no *framework*, utilizando usuários modeladores com e sem conhecimento em programação, simulando os casos de uso aqui apresentados.

Capítulo 6

Conclusão e trabalhos futuros

As arboviroses ainda são um grande problema de saúde pública do Brasil e do mundo. As mesmas são consideradas endêmicas em alguns países e presentes em diversos continentes, com exceção da Antártida. O uso de tecnologias e aproveitamento do poder computacional existente atualmente são formas de ajudar na predição, prevenção e tomadas de decisão quanto a futuras ou existentes epidemias. As ferramentas existentes conseguem atender diversos aspectos e já causam impacto, como demonstrado nos capítulos de revisão da literatura e introdução. Ainda assim, autores demonstram uma falta de ferramentas mais amplas e flexíveis que consigam atender mais aspectos do processo de modelagem e simulação.

Modelagem e simulação são ferramentas importantes e poderosas para o estudo e compreensão de fenômenos ou comportamentos. Epstein (2008) apresenta em um de seus trabalhos as motivações para se usar a modelagem, e demonstra uma série de outras possibilidades que temos ao fazer uso de modelos. Um modelo pode ser servir tanto para predição de acontecimentos, como auxiliar no planejamento e tomada de decisão quanto a determinado assunto, no levantamento de novas questões sobre o objeto de estudo e explicação melhor de algum evento.

As ferramentas de modelagem e simulação visam auxiliar o usuário em diversas tarefas: parametrização, construção de código, simulação, apresentação de saída entre outras funções. Essas ferramentas existem para os mais diversos campos de estudo e possuem suportes variados. Baseado em tudo que estas ferramentas oferecem e nas necessidades dos modeladores encontradas na literatura, foi proposta a criação de um *framework* capaz de englobar funções que deem suporte ao processo de modelagem e simulação de arboviroses. Além disso, o *framework* oferece um fluxo de uso para a ferramenta e um guia de como estrutura um software baseado na mesma. O *framework* foi construído baseado nas ferramentas de simulação já existentes encontradas

na literatura, visando oferecer melhorias neste campo e auxiliar o usuário na tarefa de modelagem e simulação. Como vimos na revisão bibliográfica diversas ferramentas de diferentes áreas de estudo compartilham funcionalidades semelhantes, isso abre a possibilidade de criar algo que seja genérico e consistente para ferramentas de modelagem.

O *framework* foi construído em módulos, cada módulo tendo independência em relação aos outros, recebendo uma entrada e saída de suas ligações e contendo uma série de *features*. Além de oferecer a arquitetura de uma ferramenta de modelagem e simulação, o *framework* funciona como um passo a passo para modeladores. Cada parte tem seu papel e momento de uso, como demonstrado nos casos de uso da ferramenta.

Um dos principais objetivos do *framework* é permitir a criação de ferramentas que sejam úteis, flexíveis, intuitivas e poderosas para modelagem e simulação de arboviroses. Como apresentado, não é possível criar uma ferramenta genérica, pois cada área de estudo tem suas particularidades, sejam elas as plataformas de simulação/interpretadores que suportam aqueles tipos de modelos ou necessidade de suporte a tipos de dados bem específicos para parametrizar o modelo. Com o *framework*, desenvolvedores podem criar suas próprias ferramentas, sem necessidade de se preocupar com sua modelagem ou decidir sobre qual arquitetura conceitual utilizar. Neste contexto, seria possível criar um documento completo descrevendo uma versão do *framework* para uma dada área de estudo de arboviroses, por exemplo, poderíamos aqui criar uma ferramenta que foque em simular modelos de Dengue, como o DengueME.

Foi necessário modificar vários aspectos da ferramenta DengueME para aproximá-la da estrutura proposta para o *framework*. Foi feita uma reengenharia e melhoria de seu fonte e documentação. A ferramenta agora tem uma nova versão com modificações que facilitam a implementação de extensões, que poderão incorporar novos recursos que outras ferramentas de modelagem não têm ou não conseguem adicionar para atender à demanda dos seus usuários.

Como demonstrado no comparativo dos casos de uso, diversas funções não estão presentes nas ferramentas levantadas aqui no trabalho. Foi estabelecida uma ligação entre o *framework* e as etapas do processo de modelagem. Essa ligação demonstra em qual etapa cada módulo pode impactar de maneira positiva. Ao montar os modelos dentro da ferramenta DengueME, que já reflete partes do *framework*, agilizamos e tornamos o processo de modelagem e simulação mais dinâmico.

O *framework* impõe alguns limitantes às ferramentas que desejem utilizá-lo como base. Por exemplo, por mais que alguns módulos possam ser condensados em um só, idealmente as ligações entre os módulos devem ser mantidas, a fim de não ferir o ciclo do *framework* e de manter suas propriedades. Outro problema é a necessidade de moldar o fonte do modelo seguindo algum padrão imposto pela ferramenta. Como é

impossível gerar fonte de forma genérica, e cada linguagem tem suas particularidades para o funcionamento da injeção de dados, a introdução de suporte a novas linguagens ou bibliotecas pode ser difícil ou até mesmo inviável. A exemplo da linguagem R, que conta com diversos packages, é necessário um mecanismo que instale os pacotes necessários para um dado modelo caso o interpretador de R na máquina do usuário não os tenha. Escrever tal código não é simples e demanda entendimento do sistema de pacotes de bibliotecas de R e de como realizar essa checagem/download.

A implementação da integração com dados geográficos é outro desafio. É necessário que o desenvolvedor tenha conhecimento em SIG e suas operações para implantar as funções mais utilizadas para modelos do objeto de estudo que sua ferramenta deseja cobrir. O *framework* descreve uma ferramenta genérica de modelagem que como dito anteriormente pode ser específica em diferentes camadas. Por exemplo, é possível gerar a partir dele uma ferramenta geral para modelagem de doenças infecciosas, e se desejar especificar mais é possível gerar uma ferramenta que foca em modelos de Arboviroses Tropicais. Podemos especificar mais ainda gerando uma ferramenta para modelos de Febre Amarela. Conceber o estudo específico que se deseja, verificando quais os modelos mais comuns dessa área, quais suas entradas e que tecnologias usar, é uma tarefa para os desenvolvedores da ferramenta. Neste contexto, o *framework* cria um caminho para aqueles que desejem criar uma ferramenta de modelagem para facilitar e fomentar sua área de estudo, mas gera a necessidade de estudos aprofundados da área pelo mesmo.

Os requisitos tecnológicos para que tudo o que foi proposto para o *framework* funcione existem em diversas linguagens de programação. Não foi realizado um estudo específico quanto a isso mas é provável que seja possível criar inclusive uma ferramenta Web que faça uso do *framework*. A tecnologia foco nesta dissertação foi o QT, que foi usado na construção de ferramentas como DengueME e QGIS, sendo estes exemplos de seu potencial. QT conta com todo o ferramental necessário para implementação dos módulos do *framework* e possibilita que desenvolvedores criem as ferramentas com toda flexibilidade e funcionalidades necessárias.

Os casos de usos foram a forma mais clara que encontramos de demonstrar o *framework* em ação e mostrar como o usuário se aproveitaria dele. Mesmo DengueME não tendo todas as telas necessárias, foi possível demonstrar pelos *mockups* qual seria a ideia daquela funcionalidade em ação. Além disso, com o caso de uso fica bem claro o que é e o que não é possível fazer na ferramenta. As ferramentas atuais não têm a mesma flexibilidade e quantidade de recursos como os apresentados em ambo os casos de uso, o que é um diferencial deste *framework*.

Um ponto interessante do *framework* é que ferramentas já existentes, como por

exemplo o NetLogo, podem fazer uso dele para incrementar mais a ferramenta e trazer outras funcionalidades ao usuário. Sendo esta uma ferramenta específica para modelagem em agentes, basta que foquem neste tema e façam com que cada um dos módulos seja útil para modeladores que utilizem este paradigma.

Como trabalho futuro está programada a finalização da ferramenta DengueME, que está em andamento até a data da entrega deste trabalho, implementando todos módulos descritos no *framework*. Módulos como o de calibragem, analisador de resultados, módulo de simulação e editor de dados estão em construção e planejamento para a ferramenta DengueME. A reengenharia da ferramenta que permite a implementação dos módulos e a criação de novas extensões esta finalizada, permitindo que outras pessoas colaborem com o trabalho em andamento. O foco do trabalho foi o estudo das ferramentas de modelagem e simulação existentes e a criação do *framework*.

Tornar público este *framework* de forma a divulgar o mesmo para que outros desenvolvedores criem ferramentas baseadas nele é outro objetivo futuro. Um teste com usuários utilizando uma ferramenta que implementa todos módulos da ferramenta seria interessante para avaliação de possíveis melhorias no mesmo. A avaliação deveria ser sobre os módulos e suas funcionalidades genéricas, eliminando qualquer *feature* específica a ferramenta que se baseou no esqueleto. Ideias de novos módulos podem surgir dessa interação com usuários, ou, até mesmo, eliminação de módulos que estejam obsoletos.

Em resumo, o trabalho tem como objetivo ser uma contribuição que venha facilitar o trabalho de desenvolvedores de ferramentas de modelagem e simulação e dar assistência aos agentes de saúde pública que precisam de ferramental para planejamento e aprofundamento em seus estudos.

6.1 Ameaças à Validade

Dentre as limitações deste trabalho podemos citar o fato da ferramenta DengueME não ter sido finalizada, sendo usado em seu caso de uso diversos *mockups* para representar as telas não finalizadas.

Neste contexto, ambos os casos de uso foram baseados na ferramenta DengueME em seu estado atual, incompleta. Sendo assim, é possível ter uma noção de como seria o uso da ferramenta com todo o *framework* implementado mas não podemos demonstrar o real impacto da ferramenta com usuários reais. Uma das limitações deste caso de uso é o fato que o mesmo foi executado pelo autor desta dissertação, mesmo sendo este experiente com modelagem, não é possível refletir todos pontos positivos e negativos

partindo de um teste.

Durante a reengenharia da ferramenta foi possível notar que algumas partes do *framework*, como o Editor de Dados e Calibrador são muito mais complexas que inicialmente imaginados. O editor de dados que serve como um pre-processamento aos dados precisa de uma atenção especial sobre quais operações oferecer e como explicar ao usuário o uso das mesmas. O calibrador precisa de especialistas na área para se decidir quais algoritmos e/ou técnicas de calibragem se usar, nem sempre a calibragem é necessária e em alguns casos uma análise dos resultados já se faz suficiente para entender como ajustar o modelo, por exemplo via análise do Tipping Point de uma variável, chegando a conclusão de qual valor causa maior impacto no modelo para aquela variável fixa.

O *framework* conceitual foi concebido como um *framework* genérico para ferramentas de modelagem no geral. Neste trabalho, o foco foi uma ferramenta de modelagem de arboviroses, DengueME, sendo assim necessário realizar entender como o *framework* funcionaria em uma ferramenta desta. Sendo está a primeira vez que o *framework* é aplicado em alguma ferramenta, existem modificações que poderiam ser eliminadas ou não realizadas e ainda assim o resultado final seria obtido. Cada um dos módulos do *framework* deve ser repensado ao ser aplicado ele em uma ferramenta para um certo tipo de modelagem.

O trabalho teve como objetivo principal a apresentação do *framework* conceitual e demonstrar de maneira pratica uma aplicação do mesmo. Ainda assim, existe grande espaço para trabalhos futuros e melhorias da concepção do mesmo.

Referências Bibliográficas

- Baianu, P. D. I. (2011). *Complexity, Emergent Systems and Complex Biological Systems: Complex Systems Theory and Biodynamics*. PediaPress: Mainz, Germany.
- Baker, E. L.; Friede, A.; Moulton, A. D. & Ross, D. A. (1995). Information network for public health officials (inpho): a framework for integrated public health information and practice. *Journal of public health management and practice: JPHMP*, 1(1):43--47.
- Basu, S. & Andrews, J. (2013). Complexity in mathematical models of public health policies: a guide for consumers of models. *PLoS medicine*, 10(10):e1001540.
- Brooks, R. & Robinson, S. (2000). *Simulation studies: Key stages and processes*. Hampshire, UK: Palgrave Macmillan.
- Bui, Q.-T.; Nguyen, Q.-H.; Pham, V. M.; Pham, M. H. & Tran, A. T. (2018). Understanding spatial variations of malaria in Vietnam using remotely sensed data integrated into GIS and machine learning classifiers. *Geocarto International*, pp. 1--15.
- Burns, I.; Semmens, D.; Scott, S.; Levick, L.; Goodrich, D. & Kepner, W. (2004). The automated geospatial watershed assessment (agwa) tool enhanced capabilities of just released version 1.4 1705. Em *Proceedings Arizona Hydrological Society Symposium*, pp. 5--18.
- Carley, K. M. (1996). Validating computational models. *Paper available at <http://www.casos.cs.cmu.edu/publications/papers.php>*.
- Carroll, L. N.; Au, A. P.; Detwiler, L. T.; Fu, T.-c.; Painter, I. S. & Abernethy, N. F. (2014). Visualization and analytics tools for infectious disease epidemiology: a systematic review. *Journal of biomedical informatics*, 51:287--298.

- Carvalho, M. S.; de Fátima Sá Pina, M. & dos Santos, S. M. (2000). Conceitos básicos de sistemas de informação geográfica e cartografia aplicadas a saúde. Em *Conceitos básicos de sistemas de informação geográfica e cartografia aplicadas a saúde*, pp. 117--117.
- Chen, D.; Moulin, B. & Wu, J. (2014). *Analyzing and Modeling Spatial and Temporal Dynamics of Infectious Diseases*. Wiley.
- Coppock, J. T. & Rhind, D. W. (1991). The history of GIS. *Geographical information systems: Principles and applications*, 1(1):21--43.
- de Lima, T.; Lana, R.; de Senna Carneiro, T.; Codeço, C.; Machado, G.; Ferreira, L.; de Castro Medeiros, L. & Davis Junior, C. (2016). DengueME: A tool for the modeling and simulation of dengue spatiotemporal dynamics. *International journal of environmental research and public health*, 13(9):920.
- Delgado-Petrocelli, L.; Ramos, S.; Gordon, E.; Martínez, N.; Montiel, E.; Zoppi, E. & Moser, J. (2004). New perspectives in public health management: The particular case of malaria. *Revista de la Facultad de Ingeniería*, 19:51--61.
- Dom, N. C.; Ahmad, A. H.; Latif, Z. A. & Ismail, R. (2017). Integration of GIS-based model with epidemiological data as a tool for dengue surveillance. *EnvironmentAsia*, 10(2).
- Donalisio, M. R.; Ribas, R.; Ii, F.; Paula, A.; Von, B. & Ii, Z. (2017). Arboviroses emergentes no Brasil: desafios para a clínica e implicações para a saúde pública. *Revista de Saude Pública*, 51(30):10--15.
- Epstein, J. M. (2008). Why Model? *Journal of Artificial Societies and Social Simulation*, 11(4).
- Farinelli, E. C.; Baquero, O. S.; Stephan, C. & Chiaravalloti-Neto, F. (2018). Low socioeconomic condition and the risk of dengue fever: a direct relationship. *Acta tropica*, 180:47--57.
- Ferreira, L. S. (2017). DengueME: um framework de software para modelagem da dengue e seu vetor.
- Ferreira, L. S; HEILMAN, P. (2015). Evaluation of GIS open source frameworks to reproduce the agwa watershed modeling tool. *LATIN AMERICAN SUMMER RESEARCH PROGRAM 2015*.

- Fletcher-Lartey, S. M. & Caprarelli, G. (2016). Application of GIS technology in public health: Successes and challenges. *Parasitology*, 143(4):401--415.
- Focks, D. A.; Haile, D.; Daniels, E. & Mount, G. A. (1993). Dynamic life table model for *Aedes aegypti* (diptera: Culicidae): analysis of the literature and model development. *Journal of medical entomology*, 30(6):1003--1017.
- Friede, A.; Blum, H. L. & McDonald, M. (1995). Public health informatics: How information-age technology can strengthen public health. *Annual Review of Public Health*, 16(1):239--252. PMID: 7639873.
- Goodrich, D. C.; Guertin, D. P.; Burns, I. S.; Nearing, M. A.; Stone, J. J.; Wei, H.; Heilman, P.; Hernandez, M.; Spaeth, K.; Pierson, F. et al. (2011). Agwa: the automated geospatial watershed assessment tool to inform rangeland management. *Rangelands*, 33(4):41--48.
- Gould, E.; Pettersson, J.; Higgs, S.; Charrel, R. & de Lamballerie, X. (2017a). Emerging arboviruses: why today? *One Health*, 4:1--13.
- Gould, E.; Pettersson, J.; Higgs, S.; Charrel, R. & de Lamballerie, X. (2017b). Emerging arboviruses: Why today? *One Health*, 4:1--13.
- Griffin, J. T.; Hollingsworth, T. D.; Okell, L. C.; Churcher, T. S.; White, M.; Hinsley, W.; Bousema, T.; Drakeley, C. J.; Ferguson, N. M.; Basanez, M.-G. et al. (2010). Reducing plasmodium falciparum malaria transmission in Africa: a model-based evaluation of intervention strategies. *PLoS medicine*, 7(8):e1000324.
- Gubler, D. J. (2002). The global emergence/resurgence of arboviral diseases as public health problems. *Archives of medical research*, 33(4):330--342.
- Guo, J.-G.; Vounatsou, P.; Cao, C.-L.; Utzinger, J.; Zhu, H.-Q.; Anderegg, D.; Zhu, R.; He, Z.-Y.; Li, D.; Hu, F.; Chen, M.-G. & Tanner, M. (2005). A geographic information and remote sensing based model for prediction of oncomelania hupensis habitats in the Poyang lake area, China. *Acta tropica*, 96(2):213--222.
- Haddad, H.; Moulin, B. & Theriault, M. (2016). A fully GIS-integrated simulation approach for analyzing the spread of epidemics in urban areas. *SIGSPATIAL Special*, 8(1):34--41.
- Higa, Y. (2011). Dengue vectors and their spatial distribution. *Tropical medicine and health*, 39:S17--S27.

- Higgs, G. (2004). A literature review of the use of GIS-based measures of access to health care services. *Health Services and Outcomes Research Methodology*, 5(2):119-139.
- Horta, M. A. P.; Ferreira, A. P.; de Oliveira, R. B.; Wermelinger, E. D.; de Oliveira Ker, F. T.; Ferreira, A. C. N. & Catita, C. M. S. (2014). Os efeitos do crescimento urbano sobre a dengue. *Revista Brasileira em Promoção da Saúde*, 26(4):539-547.
- Jacobson, M.; Taylor, C.; Richards, D. & Lai, P. (2013). Computational scientific inquiry with virtual worlds and agent-based models: New ways of “doing” science to learn science.
- Karl, S.; Halder, N.; Kelso, J. K.; Ritchie, S. A. & Milne, G. J. (2014). A spatial simulation model for dengue virus infection in urban areas. *BMC infectious diseases*, 14(1):447.
- Kermack, W. O. & McKendrick, A. G. (1991). Contributions to the mathematical theory of epidemics. 1927.
- Kilpatrick, A. M. & Randolph, S. E. (2012). Drivers, dynamics, and control of emerging vector-borne zoonotic diseases. *The Lancet*, 380(9857):1946-1955.
- Kirby, R. S.; Delmelle, E. & Eberth, J. M. (2017). Advances in spatial epidemiology and geographic information systems. *Annals of Epidemiology*, 27(1):1-9.
- Kong, C. Y.; McMahon, P. M. & Gazelle, G. S. (2009). Calibration of disease simulation model using an engineering approach. *Value in health*, 12(4):521-529.
- Kuno, G. & Chang, G.-J. J. (2005). Biological transmission of arboviruses: reexamination of and new insights into components, mechanisms, and unique traits as well as their evolutionary trends. *Clinical microbiology reviews*, 18(4):608-637.
- Lana, R. M.; Morais, M. M.; de Lima, T. F. M.; de Senna Carneiro, T. G.; Stolerman, L. M.; Santos, J. P. C. D.; Cortés, J. J. C.; Eiras, Á. E. & Codeço, C. T. (2018). Assessment of a trap based *Aedes aegypti* surveillance program using mathematical modeling. *PloS one*, 13(1):1-16.
- London, I. C. (2010). Malaria modelling - tools and data. <https://www.imperial.ac.uk/malaria-modelling/tools-and-data/>.
- Longley, P. A.; Goodchild, M. F.; Maguire, D. J. & Rhind, D. W. (2005). *Geographic information systems and science*. John Wiley & Sons.

- Ministério da Saúde, (MS) Sistema de Vigilância em Saúde, S. (2017). Boletim epidemiológico 22, 2019. <https://portalarquivos2.saude.gov.br/images/pdf/2019/setembro/11/BE-arbovirose-22.pdf>.
- Ng, V.; Fazil, A.; Gachon, P.; Deuymes, G.; Radojević, M.; Mascarenhas, M.; Garasia, S.; Johansson, M. A. & Ogden, N. H. (2017). Assessment of the probability of autochthonous transmission of chikungunya virus in Canada under recent and projected climate change. *Environmental health perspectives*, 125(6):067001.
- Ostfeld, R. S.; Glass, G. E. & Keesing, F. (2005). Spatial epidemiology: An emerging (or re-emerging) discipline. *Trends in Ecology and Evolution*, 20(6):328--336.
- Park, B. & Qi, H. (2005). Development and evaluation of a procedure for the calibration of simulation models. *Transportation Research Record*, 1934(1):208--217.
- Patterson, J.; Sammon, M. & Garg, M. (2016). Dengue, Zika and Chikungunya: Emerging Arboviruses in the New World. *Western Journal of Emergency Medicine*, 17(6):671--679.
- Rao, D. M. (2016). Efficient parallel simulation of spatially-explicit agent-based epidemiological models. *Journal of Parallel and Distributed Computing*, 93-94(1):102--119.
- Rao, D. M.; Chernyakhovsky, A. & Rao, V. (2006). Searums: studying epidemiology of avian influenza rapidly using modeling and simulation. *Lecture Notes in Engineering and Computer Science*, 2167.
- Riehle, D. (2000). Framework design: A role modeling approach. *Softwaretechnik-Trends*, 20.
- Riley, S.; Eames, K.; Isham, V.; Mollison, D. & Trapman, P. (2015). Five challenges for spatial epidemic models. *Epidemics*, 10:68--71.
- Rust, R. S. (2012). Human Arboviral Encephalitis. *Seminars in Pediatric Neurology*, 19(3):130--151.
- Schoolfield, R.; Sharpe, P. & Magnuson, C. (1981). Non-linear regression of biological temperature-dependent rate models based on absolute reaction-rate theory. *Journal of theoretical biology*, 88(4):719--731.
- Shaw, N. T. (2012). Geographical information systems and health: current state and future directions. *Healthcare informatics research*, 18(2):88--96.

- Smith, A.; Carter, I. et al. (1984). International transportation of mosquitoes of public health importance. *Commerce and the spread of pests and disease vectors.*, pp. 1--21.
- Smitha, D. L.; Perkin; Alex, T.; Reiner, R. C.; Barker, C. M.; Niu, T.; Chaves, L. F.; Ellis, A. M.; George, D. B.; Menach, A. L.; Pulliam, J. R.; Bisanzio, D.; Buckee, C.; Chiyaka, C.; Cummings, D. A.; Garcia, A. J.; Gatton, M. L.; Gethingv, P. W.; Hartley, D. M.; Johnston, G.; Klein, E. Y.; Michael, E.; Lloyd, A. L.; Pigott, D. M.; Reisen, W. K.; Ruktanonchai, N.; Singh, B. K.; Stoller, J.; Tatem, A. J.; Kitron, U.; Godfray, H. C. J.; Cohen, J. M.; Hay, S. I. & Scott, T. W. (2014). Recasting the theory of mosquito-borne pathogen transmission dynamics and control. *Transactions of the Royal Society of Tropical Medicine and Hygiene*, 108(4):185--197.
- Sokolowski, J. & Banks, C. (2010). *Modeling and Simulation Fundamentals: Theoretical Underpinnings and Practical Domains*. Wiley.
- Stout, N. K.; Knudsen, A. B.; Kong, C. Y.; McMahon, P. M. & Gazelle, G. S. (2009). Calibration methods used in cancer simulation models and suggested reporting guidelines. *Pharmacoeconomics*, 27(7):533--545.
- Tisue, S. & Wilensky, U. (2004). Netlogo: A simple environment for modeling complexity. pp. 16--21.
- Weber, S. & Weber, S. S. (2004). *The Success of Open Source*. Harvard University Press.
- Wilder-Smith, A.; Gubler, D. J.; Weaver, S. C.; Monath, T. P.; Heymann, D. L. & Scott, T. W. (2017a). Epidemic arboviral diseases: priorities for research and public health. *The Lancet infectious diseases*, 17(3):101--106.
- Wilder-Smith, A.; Gubler, D. J.; Weaver, S. C.; Monath, T. P.; Heymann, D. L. & Scott, T. W. (2017b). Epidemic arboviral diseases: priorities for research and public health. *The Lancet infectious diseases*, 17(3):e101--e106.
- Zeilhofer, P.; Arraes Neto, P.; Maja, W. & Vecchiato, D. (2009). A web-based, component-oriented application for spatial modelling of habitat suitability of mosquito vectors. *International Journal of Digital Earth*, 2(4):327--342.