Análise do LUCC-ME

Versão 1: Pedro, 06 de janeiro de 2011

Este relatório analisa a implementação do pacote LUCC-ME e de suas funções individuais. Ele está dividido da mesma forma que a estrutura de diretórios do pacote.

O que está em verde são comentários do Pedro como desenvolvedor do TerraME.

Comentários Gerais

1. O cabeçalho de cada arquivo do pacote deve conter o nome e a versão do software e os seus autores. Não precisa colocar a licença GNU toda em cada arquivo, basta colocar o arquivo no diretório raiz do pacote e dizer em cada cabeçalho que a licença segue junto com o pacote. Esta é a maneira padrão de distribuição de software livre.
2. Traduzir todos os nomes das funções e comentários para o inglês. Alguns nomes de função estão metade em inglês e metade em português.
3. O código está mal indentado, levando a interpretações erradas para quem está analisando o mesmo. Por exemplo, na linha 29 do arquivo LogisticRegressionModel.lua, o “end” está terminando um “if”, e não um “for” como parece ser. Verificar e corrigir toda a indentação do LUCC-ME.
4. Seria interessante ter os arquivos com o mesmo nome da classe que é implementada dentro deles. Isto tornaria mais fácil a localização do código.
5. Mensagens de erro devem usar a função “error”. Esta função recebe dois parâmetros, uma string com a descrição do erro e um número com a quantidade de elementos que devem ser removidos da pilha de execução antes de o erro executar. Este número serve para fazer com que o erro apareça na linha que chamou a função ao invés de ocorrer dentro da própria função. Por exemplo `error("Filename not defined.", 2)`.
6. Em vários pontos do LUCC-ME, os parâmetros “self” e “model” se confundem. Por exemplo, no arquivo Demand.lua, em `model.getDemand = function (self,model,time,uso)`, “self” é o mesmo objeto que o “model” do “model.getDemand”, mas “model”, o segundo argumento, representa na verdade um “LandChangeModel”. Usar o nome model para duas coisas distintas deixa o código muito confuso. Por que não chamar o segundo “model” de “landchangemodel”, ou algo similar?
7. O LUCC-ME poderia ter uma variável global booleana USE_LOG (ou algo parecido) para mostrar ou não mensagens de acompanhamento do que está acontecendo na simulação.

Projects

8. Aparentemente este diretório pode ser apagado do LUCC-ME, pois possui apenas um arquivo prj, que não é usado pelo TerraME.

Region

9. A princípio, este diretório não deveria existir, pois a única função que ele implementa é completamente dependente de uma base de dados específica, usando atributos e datas predeterminadas.

TerraME_I

10. Este diretório não deveria existir, uma vez que o código do TerraME não deveria ser disponibilizado junto com o pacote.

Save

Na função de salvamento:

11. a variável “landUses” não é usada.
12. esta função poderia ser implementada de forma a gerar eventos únicos para cada um dos anos, ao invés de ter que percorrer todo o vetor de tempos uma vez para cada tempo. Ver discussão do item 19.
13. Dois parametros seriam interessante de se ter no save: “yearly” e “onlyAtEnd”, sendo os dois booleanos, indicando se o espaço celular deve ser salvo ao final de cada ano, ou se apenas uma vez, ao final da execução do modelo, respectivamente. Estes seriam os dois usos mais comuns, e caso o modelador não indique explicitamente os tempos, um destes dois poderia ser usado.

luccFramework

14. Por que um modelo precisa ter necessariamente um “name”, senão o “LandChangeModel” aborta?

Pedro: O TerraME usa um id único para seus objetos, alguns deles são obrigatórios para o modelador, mas vou discutir com Tiago para ver se isto não pode ficar escondido caso o modelador não precise.

15. “LandChangeModel” pode ser um bom nome de classe caso ela seja extensível também para modelos bottom-up. Isto será possível? Caso contrário, o termo top-down ou algo parecido deveria aparecer no nome desta classe.
16. Fazer código para verificar se “regressionData”, “annualDemand” e “allocationData” possuem o mesmo tamanho de “landUseTypes” e mostrar mensagens de erro caso necessário.
17. Verificar se cada um dos “LandUseDrivers” e “landUseTypes” descritos pelo modelador são atributos das células do espaço celular (deve ser feito após o load), para evitar erros no decorrer dos modelos de potencial e alocação.
18. Seria correto dizer que os atributos a serem lidos do espaço celular são apenas os “landUseTypes” e “landUseDrivers”? Se for, esta tarefa poderia ser automatizada antes e fazer o load do banco de dados para economizar memória. Isto pode evitar ter que implementar o item anterior. Parece que o “region” também é usado. Ver item 28.3.
19. O “LandChangeModel” poderia retornar um Environment, já com o Timer pronto, e não deixar para o modelador fazer isto. Na verdade, cada um dos componentes (potencial, demanda e save, poderiam cada um retornar um vetor de eventos que serão executados. O “LandChangeModel” poderia agregar todos estes vetores de eventos em um único Timer e retorná-lo. Assim, um potencial que gerasse valores apenas no primeiro ano geraria apenas um evento, ou tantos eventos quanto sejam os dados disponíveis para a atualização do potencial; uma alocação geraria um Event para cada ano disponível de demanda, assim como o “databaseSave”. As prioridades de cada um dos três (potencial,

alocação e save) poderia ser pré-estabelecida, com $\text{prioridade}(\text{potencial}) > \text{prioridade}(\text{alocação}) > \text{prioridade}(\text{save})$. Assim, o modelador poderia inserir novos eventos no Timer de forma que fosse possível executar qualquer coisa entre cada um dos sub-componentes do modelo, tornando a ferramenta mais flexível.

20. Existe uma repetição do tempo em "LandChangeModel" (timeStart e timeEnd) assim como no "Demand4Scenarios" (initialTime e finalTime). Será que o tempo do "LandChangeModel" não poderia ser removido, ficando o tempo ligado apenas à demanda? Para que o potencial precisaria de tempo inicial e final?
21. O Lucc-ME atualmente considera o intervalo de tempo como sendo sempre 1. Este poderia ser um valor default, mas deveria ficar a cargo do usuário.

Pedro: O TerraME precisa melhorar a representação do tempo. Quando a escala de tempo é anual não existe qualquer problema usar números para representar os anos, mas alguns modelos de mudanças climáticas possuem intervalo de tempo de meia hora, o que tornaria o usuário responsável pelos cálculos de dias e anos. Acho que a questão temporal do Lucc-ME deve ser melhorada apenas quando a representação de unidades de tempo do TerraME for implementada.

22. Quando se tem apenas uma "region" nos modelos Lucc, deveria ser possível evitar o uso de um escopo a mais ("{}"), por exemplo, em "transitionMatrix" de "AllocationCluesLike" e "regressionData" de "LogisticRegressionModel". Isto é uma estratégia usual na implementação atual do TerraME. Por exemplo, na construção do CellularSpace, o parâmetro select usa um vetor de strings, mas pode ser apenas uma string ao invés de um vetor com uma única string dentro.
23. Remover os +1 dos índices de vetores em Lua. Índices de vetor em Lua começam em 1 e não em 0. Local: LinearRegression.lua linha 182. Verificar se existe esta indexação errada em mais algum lugar.
24. Fazer verificação sobre se é possível acoplar os sub-modelos que estão sendo usados pelo "LandChangeModel" (discreto e contínuo, por exemplo), e retornar mensagem de erro caso o modelado esteja acoplando modelos que não são compatíveis.
25. Do mesmo modo que todas as classes do Lucc-ME possuem um "execute(...)", elas poderiam ter um "verify(...)" para fazer as verificações necessárias para os valores dos parâmetros usados.

Potential

26. fazer código para verificar, em "regressionData", se betas tem o mesmo tamanho de "attributes" e mostrar mensagem de erro caso necessário. Verificar também se os elementos de "attributes" são índices válidos dentro do vetor de "landUseDrivers".
27. em "LinearRegression", a regressão é aplicada para cada célula do espaço celular. Verificar se este cálculo da regressão demanda bastante tempo, pois ele pode ser otimizado significativamente, removendo-se um for, da mesma forma que o novo código do synchronize.
28. Em LogisticRegressionModel.lua:
 - 28.1. O comentário da linha 15 precisa ser executado sim, mas este não é o lugar mais recomendado para fazer isto. Da mesma forma, o "if" da linha 48 precisa ser executado, mas no local que ele está é ineficiente.

- 28.2. Aparentemente, a regressão logística é constante durante toda a execução do modelo, uma vez que os drivers não são alterados. Desta forma, a regressão deveria ser feita apenas uma vez, com o execute atualizando apenas a “suit[ability]_” por causa da elasticidade.
- 28.3. “region”, um atributo das células, deveria ser um atributo declarado no “LandChangeModel” diretamente. Isto deixaria livre para o modelador escolher o atributo que quiser, além de facilitar a documentação. A verificação da disponibilidade de diferentes regressões para cada região deveria ser feita antes de calcular as regressões.
29. Em LogisticRegressionModelNeighbourhood.lua:
- 29.1. Linha 23: CreateMooreNeighborhood basta ser criado apenas uma vez, e não toda vez que este potencial é calculado.
- 29.2. Linha 36: Em “numNeigh <= 3”, esse 3 deveria ser um parâmetro do modelo, e não um valor constante para ativar a elasticidade. Adicionalmente, este if e o próximo elseif deveriam estar depois do forEachNeighbor.
30. modelo bayse2.lua deve ser apagado.
31. Em LinearRegression.lua, “adaptRegressionConstants” não deveria pertencer ao arquivo AllocationClueLike.lua? Esta atualização é dependente do modo de alocação feito pelo CLUE.

Demanda

32. “cellArea” tem que ser uma variável cujo valor deve ser lido do banco, e não criado dentro do espaço celular pelo modelador.
- Pedro: Vou tentar colocar isto na próxima versão do TerraME. Adicionalmente, o TerraME precisa de definições padrões para medidas de tempo e espaço. Por exemplo, as unidades padrão para tempo e espaço poderiam ser anos e quilômetros, respectivamente. Assim, quando se for usar valores referentes a estas métricas a ferramenta usaria estas unidades de medida, por exemplo para ler a área das células do banco. Da mesma forma que no item 21, o TerraME precisa de uma representação de unidades de medida do espaço, por exemplo para dizer que todos os dados relativos a distância e área serão medidos em km, ha, tarefa, etc., de forma a padronizar seu o uso e ler do banco já na unidade de medida predefinida.
33. “anualDemand” não poderia ser apenas a quantidade de mudança, ao invés do valor total de cobertura de um determinado tipo? Ficaria muito mais legível, valores positivos indicariam um crescimento do uso, negativos indicariam uma redução e zeros indicariam que a cobertura não mudou. Desta forma, a soma de uma linha deveria ser sempre zero, ao invés de ser o tamanho do espaço celular, como é hoje (o que pode levar a erro caso exista alguma mudança no número de células no banco de dados). Outra possibilidade seria fazer uma classe nova para a descrição da demanda desta forma, deixando para o modelador escolher como ele quer descrever a demanda.
34. Não ficou claro se a primeira linha da demanda se refere ao dado atual lido do banco ou se ela é realmente a primeira alteração a ser feita no espaço celular. Da mesma forma, o “initialTime” se refere ao primeiro tempo de mudança ou ao tempo do dado que foi lido do banco de dados? O mesmo se aplica ao finalTime. Estas questões devem ser bem explicadas na documentação.

35. "ComputeDemand" não é um nome bom para classe. O uso de verbos é aconselhável apenas no nome de funções.
36. o LUCC-ME funcionaria se eu tiver uma demanda "ComputeDemand", por exemplo, para os anos de 2000-2010, e depois eu quisesse aplicar no mesmo modelo uma demanda fictícia como um cenário para anos futuros? Ou eu teria que transformar isto em dois modelos?
37. Em "Demand4Scenarios", verificar se o número de linhas da demanda é pelo menos "finalTime – initialTime", mostrando mensagem de erro caso contrário.
38. As funções de demanda seriam bem mais eficientes se elas processassem os conjuntos de dados que elas possuem e gerassem uma tabela indexada pelo ano e pelo tipo de uso, de forma que, durante a execução do modelo, as demandas fossem acessadas rapidamente. Por exemplo, o "ComputeDemand" poderia ser otimizado para não ter que calcular a demanda toda vez que o modelo fosse executado. Ele poderia gerar um arquivo de demanda na primeira vez, e então tentar ler este arquivo para evitar ter que recalculá-la em execuções sucessivas.

Allocation

39. A atual implementação usa uma série de funções auxiliares. O problema é que todas estas funções são declaradas como variáveis locais, e podem gerar conflito no futuro quando as implementações começarem a crescer. Aliás, em diferentes arquivos existem declarações de funções com o mesmo nome (toIndex, changeUse, calcDifferences, talvez outras mais): estas funções são sobrescritas, e apenas a última a ser carregada é que realmente será usada pelo LUCC-ME. É melhor usar uma estrutura de classes, assim com o TerraME usa. Ver a implementação do arquivo TerraME.lua.
40. em AllocationCluesLikePatchChange.lua:
 - 40.1. linha 189: não deveria criar uma vizinhança de Moore internamente, ele deveria pedir apenas um índice de vizinhança, que deveria ser criada pelo modelador, tornando a implementação mais flexível.
 - 40.2. linha 241: uso de atributo do banco "uso99_4mec" para fazer os cálculos. Este valor deve virar pelo menos um argumento do modelo ou então ser retirado.
 - 40.3. parece que o "AllocationCluesLikePatchChange" não usa a vizinhança de Moore na alocação, e sim a uma vizinhança quadrada que é calculada durante a alocação, na função "patchChangeUse".
41. em AllocationClueLike.lua:
 - 41.1. remover os "uso6_floresta", e adicionar um parâmetro para este modelo ("Allocation4ContinuousData") com informação de qual atributo representa floresta. Na verdade, o nome do atributo não deveria ser "forest" ou algo similar, mas o tipo de uso original, que não precisa ser necessariamente floresta.
 - 41.2. tenho minhas dúvidas sobre se o synchronize é necessário nesta implementação. Ele torna o código muito lento, e a princípio não existe qualquer processo dinâmico que não consiga ser simulado sequencialmente no CLUE.
Pedro: o synchronize está sendo otimizado.
 - 41.3. Padronizar os nomes das variáveis usadas internamente pelo modelo. Em alguns trechos, tem-se 'lu.."_diff", 'lu.."aux" e em outros "'suit_"..lu'. O TerraME tem

como padrão colocar ‘_’ no final de qualquer variável de uso interno, para facilitar a identificação. Esta estratégia também poderia ser usada aqui.

- 41.4. Linha 462: Evitar usar string para representar valores de atributos, isto torna o código muito susceptível a erros. Melhor usar CONSTANTES ou números inteiros.
 - 41.5. Linha 503: MIN_ELASTICITY não foi definido anteriormente!
 - 41.6. Linha 673: “event” não é parâmetro de getDemandValue(). Ele vai tentar pegar isto de uma variável global.
 - 41.7. Linha 697: “isCoupler” não é parâmetro de countAllocatedLandUseArea(). Ele vai tentar pegar isto de uma variável global. Talvez por isto o “if” desta linha esteja fazendo a mesma coisa do “else”.
42. Dada a demanda calculada por um modelo qualquer, é possível dividir esta demanda em diferentes modelos de alocação? O Dinâmica divide porcentagens entre o patcher e o expander. Seria interessante ter estes dois modelos de alocação implementados separadamente, de forma que eles pudessem ser juntados durante a implementação do modelo LUCC.
43. Em AllocationCluesLike.lua:
- 43.1. Quando se tem uso da terra discreto, para que ter um atributo para cada uso, ao invés de ter apenas um atributo com um número indicando o uso atual?
 - 43.2. Linha 299: o que quer dizer “faciter”?
 - 43.3. Linha 399: “cellsBuffer” é desnecessário e pode ser removido.