ANÁLISE DA AQUISIÇÃO DE UMA RESIDÊNCIA EMPREGANDO SIMULAÇÃO DE MONTE CARLO

Cesar Rafael Lopes
ceslop84@gmail.com

Disciplina: **Programação Avançada** – Prof. Dr. Robson R. Linhares
Departamento Acadêmico de Informática – DAINF - Campus Curitiba
Universidade Tecnológica Federal do Paraná - UTFPR
Avenida Sete de Setembro, 3165 - Curitiba/PR, Brasil - CEP 80230-901

Resumo - Este trabalho aborda o desenvolvimento de um *software* no escopo da disciplina de Programação Avançada, visando o emprego de técnicas de programação orientada a objetos em C++. Para esta finalidade, optou-se por desenvolver uma modelagem de matemática financeira para o caso de aquisição de uma residência usando um recurso estatístico conhecido como Simulação de Monte Carlo. O fluxo do sistema foi divido em três etapas: (i) entrada de dados; (ii) cálculo da simulação, e; (iii) análise dos resultados. Na etapa de entrada de dados o usuário deve registrar no sistema os valores de diversos parâmetros e a distribuição estatística que melhor os representam, bem como sua respectiva variação. Por exemplo, o usuário deve cadastrar os valores estimados de gastos com materiais, mão de obra, impostos, taxas, financiamento, etc. e as respectivas distribuições estatísticas. O sistema então executa o cálculo de uma modelagem de matemática financeira para definir o valor mais provável que será despendido em todo o processo de aquisição, considerando um intervalo de confiança de 95%. Adicionalmente, caso parâmetros adicionais tenham sido registrados, será calculada uma análise econômica e financeira, demonstrando, como por exemplo, Taxa Interna de Retorno (TIR), Valor Presente Líquido (VPL), *payback*, Razão Benefício-Custo, etc.

Palavras Chave ou Expressõe-chave Simulação de Monte Carlo, Análise de Projetos, Viabilidade Econômica e Financeira, Programação Orientada a Objetos.

Abstract - This paper addresses the development of a software within the scope of the subject of Advanced Programming, aiming at the use of object oriented programming techniques with C ++. For this purpose it was decided to develop a financial mathematical modeling for the case of acquisition of a residence using a statistical resource known in the literature as Monte Carlo Simulation. The software's workflow was divided into three stages: (i) parameter data entry; (ii) simulation calculation, and; (iii) result analysis. In the data entry stage, the user must input the values of several parameters and the statistical distribution that best represents their respective variation. For instance, the user must input the estimated values of expenses with materials, manpower, taxes, fees, financing, etc. and the respective statistical distributions. Based on these data, the system performs the calculation of a financial mathematical modeling to define the most probable value that will be spent throughout the acquisition process, considering a 95% confidence interval. Furthermore, if some extra parameters are input, the software will calculate an economic and financial analysis, displaying, for example, the Internal Rate of Return (IRR), the Net Present Value (NPV), the payback, benefit–cost ratio (BCR), etc.

Key-words or Key-expressions Monte Carlo simulation, Project Analysis, Economic and Financial Feasibility, Object-Oriented Programming

INTRODUÇÃO

O processo de tomada de decisão de aquisição de uma residência é complexo e subjetivo. Ao longo do processo de análise diversos fatores são ponderados: localização, vizinhança próxima, material de construção da residência, insolação do imóvel (norte ou sul), etc. Contudo, um critério de decisão é bastante objetivo e racional: o preço. Considerando os preços do atual mercado imobiliário associados aos altos juros praticados pelo setor bancário brasileiro, esta é uma decisão que pode facilmente ecoar por anos e mesmo décadas na vida financeira de uma família. Contudo, observa-se que mesmo sendo uma decisão bastante objetiva, percebe-se que a falta de instrução em conhecimentos básicos de matemática financeira podem conduzir um indivíduo a uma tomada de decisão equivocada e desastrosa. Em muitas ocasiões o adquirente desconhece o impacto que a incerteza na variação dos valores da reforma pode acarretar no valor final pago pela residência. Ou ainda, o comprador subestima pequenas variações nas taxas de juros ofertadas em diferentes propostas de financiamento

e acaba tendo custos acima dos previstos inicialmente. Para todos os efeitos deste estudo, serão considerados componentes do preço final da residência todo e qualquer valor diretamente ou indiretamente apropriado no processo de aquisição, tais como: taxas cartoriais, impostos em todas as esferas, mão de obra, materiais, serviços diversos (ex: corretagem, projetos, fiscalização, etc.), custos de capital (ex: financiamento, empréstimos, hipotecas, etc), custos de oportunidade (ex: deixar de aplicar o capital em outras modalidades de investimento), etc.

O objetivo deste sistema é permitir ao usuário avaliar o investimento total e a viabilidade econômica e financeira do projeto de aquisição de uma residência. Tal análise empregará como fundamento a aplicação de uma Simulação de Monte Carlo para o cálculo estatístico do valor mais provável de todo o processo, com base nos valores de despesas e receitas registrados e suas respectivas variações de valores. Isso permitirá exibir, por exemplo, que para um conjunto de valores o comprador, com 95% de probabilidade, irá despender até R$ 300.000,00 (trezentos mil reais) para realizar o projeto de aquisição da residência pretendida. Adicionalmente o sistema desenvolvido permite o cálculo de uma análise de viabilidade econômico financeira para o projeto de aquisição da residência. Destaca-se que para que seja possível calcular esta análise, caberá ao usuário o registro de informações como custo de oportunidade (receitas auferidas caso ele aplicasse o capital da compra do imóvel em outra modalidade de investimento), custo médio de locação (pois a compra do imóvel pressupõe um benefício futuro que no caso seria não ter que pagar aluguel para fins de habitação), dentre outros critérios. Este tipo de análise é conduzida para verificar se determinado projeto é viável ou vantajoso do ponto de vista financeiro. Obviamente que esta análise considerada exclusivamente aspectos objetivos e determinísticos de um projeto, neste caso a compra de um imóvel, desconsiderando aspectos relevantes, mas, de cunho pessoal e subjetivo, conforme já abordado no início desta introdução. Este resultado é, em suma, uma síntese de indicadores que permite ao adquirente uma visão analítica do desempenho financeiro do projeto, como, por exemplo, Taxa Interna de Retorno (TIR), Valor Presente Líquido (VPL), *payback*, Razão Benefício-Custo, etc.

Para o desenvolvimento foi empregado o ciclo clássico de Engenharia de *Software* de forma simplificada, dado o tamanho do sistema. Portanto o sistema foi desenvolvido em 4 etapas, sendo elas: (i) levantamento de requisitos, (ii) análise e projeto, (iii) implementação ou codificação, e (iv) testes.

Ao longo deste documento será abordada uma explicação pormenorizada das funcionalidades e contexto do software desenvolvido, descrição do ciclo de desenvolvimento, conceitos de programação Orientada a Objetos (OO) empregados bem como aplicação de Padrões de Projeto (ou *Design Patterns*) e discussão sobre os resultados obtidos.

EXPLICAÇÃO DO SOFTWARE

Com base na problemática exposta na introdução deste trabalho, o objetivo deste sistema é permitir ao usuário uma forma simples e eficaz para avaliar a viabilidade econômica e financeira do projeto de aquisição de uma residência. Tal análise empregará como fundamento a aplicação de uma Simulação de Monte Carlo para o cálculo estatístico do valor mais provável de todo o processo, com base num determinado nível de confiança. A Simulação de Monte Carlo, ou ainda Modelagem ou Método de Monte Carlo, refere-se a uma metodologia para resolver problemas matemáticos usando amostras aleatórias [1]. Um exemplo desse conceito é o cálculo do valor de expectativa de uma variável aleatória, sendo que ao invés de calcularmos o valor desta expectativa conforme sua definição teórica, que pode envolver a resolução de integrais complexas, é realizada a coleta de uma grande quantidade de amostras aleatórias e seu valor esperado é estimado baseado nessas amostras. Segundo registros históricos [2] o nome advém do famoso Cassino de Monte Carlo, em Mônaco, tendo o método sido bastante explorado pelos físicos responsáveis pelo Projeto Manhatam durante a construção da primeira bomba atômica pelos Estados Unidos da América nos anos 40, no século passado. Corriqueiramente acadêmicos e profissionais da área de finanças, num tom jocoso, afirmam que se o método tivesse sido desenvolvido nos tempos atuais se chamaria Método de Las Vegas, ao invés da referência ao celebre cassino europeu. Essa analogia entre a metodologia estatística e um cassino não é por acaso, pois o conceito aplicado deriva da total aleatoriedade dos resultados, de forma semelhante a uma casa de jogos.

Assim, o Método de Monte Carlo será empregado nos valores de despesas cadastrados pelo usuário bem como na variação observada para o montante a ser incorporado ao projeto via capital externo (ex: financiamento, hipoteca, etc.), e ao realizar uma massiva variação aleatória destes valores, determinar com um intervalo de confiança de 95%, o valor máximo mais provável para a aquisição da residência. Digamos, por exemplo, que o usuário tenha cadastrado que a instalação de piso, no escopo de reforma da nova residência, tenha uma quantidade estimada de 50m² com um valor médio de R$ 25,00/m². Estes números nos conduzem à um cálculo bastante determinístico, ao multiplicar a quantidade pelo valor unitário, de um total de R$ 1.250,00. Todavia, ao longo da realização de qualquer projeto é comum se observar variações, tanto na estimativa de quantidade quanto na distribuição dos preços. As razões podem ser as mais diversas, apenas para citar o exemplo em questão, a estimativa de quantidade pode variar por erros na medida das áreas, por perda de material ao longo da instalação, etc. e o preço varia naturalmente, pois cotações, de praxe, são obtidas de diferentes fornecedores.

Portanto, uma abordagem mais ampla levaria o usuário a considerar o valor médio associado a uma distribuição de probabilidade de como seus valores estarão dispersos. Nesse exemplo, o usuário poderia prever 50m², sendo no máximo 60m² e o mínimo 45m², e para o preço os valores poderiam estar distribuídos numa curva normal com média de R$ 25,00/m² e desvio padrão de R$ 5,00/m². Assim, os valores finais teriam igualmente um valor mínimo e um valor máximo.

Com base neste contexto, é bastante simples determinar uma amplitude para uma única atividade, no caso instalação de piso, todavia a complexidade é sobrelevada quando tratamos de dezenas ou centenas de variáveis com diferentes distribuições de dispersão. Somando a isso ainda a aleatoriedade das diversas variáveis obtêm-se um panorama típico para a aplicação de uma Simulação de Monte Carlo. O sistema desenvolvido no âmbito deste projeto trata de um escopo bastante reduzido (compra de uma residência), contudo as premissas são igualmente aplicáveis para projetos de grande envergadura, como empreendimentos do setor elétrico (linhas de transmissão, usinas, etc.), óleo e gás (refinarias, gasodutos), dentre outros. É prática comum na concepção de planos de negócio vultosos a aplicação do Método de Monte Carlo, seja para a estimativa de valores (ex: valor total de investimento, lance máximo de leilão, etc.) seja para a estimativa de taxas (ex: *spread* de risco, Taxa Interna de Retorno, etc.). Em resumo, as limitações adotadas para este projeto de *software* foram adotadas exclusivamente visando permitir a sua implantação no prazo definido, contudo, o modelo de classes é escalável para a aplicação em situações de maior complexidade. Apenas para citar um exemplo destas limitações, que serão abordadas mais adiante, cita-se a limitação máxima de 50 registros para valores de despesa e/ou receita. Esta limitação tratou apenas de evitar o desenvolvimento de um botão para que novas linhas fossem acrescentadas à tabela, não sendo uma restrição nativa do modelo do *software*.

Assim para implementar este conceito o sistema realiza um número de iterações igual ao número de amostras definida pelo usuário a fim de variar os valores de dois parâmetros: (i) capital externo, e (ii) despesas e receitas. Estes valores quando cadastrados pelo usuário foram devidamente preenchidos com seus critérios de distribuição estatística, sendo que para o escopo deste sistema foram implementados dois tipos: (i) distribuição normal [3], e (ii) distribuição uniforme [3]. Desta forma o sistema realiza múltiplas iterações com variações aleatórias dos parâmetros pré-cadastrados conforme suas respectivas distribuições de dispersão. A literatura especializada recomenda que este número de iterações, ou amostras, seja proporcional à complexidade do fenômeno em análise. Assim, faz-se necessário para o caso desta aplicação considerar pelo menos algumas centenas de milhares de iterações, ou até mesmo alguns milhões, caso muitos valores de despesa e receita sejam cadastrados e com variações muitos significativas.

O sistema não possui nenhum tipo de acesso pessoal (usuário/senha). Ao iniciar o sistema o usuário será conduzido para a tela de cadastro dos parâmetros iniciais do modelo. Os parâmetros são categorizados em seis tipos, conforme ilustração da tela de cadastro está disponível na Figura 1: (i) Dados da transação (valor do imóvel, valor de entrada e aporte FGTS); (ii) Dados do capital externo (financiamentos, empréstimos, hipotecas, etc.); (iii) Dados de despesas (materiais, mão de obra, taxas, impostos, outras despesas, etc.) e eventuais receitas (venda de bens como, por exemplo, venda de móveis que vieram com a nova residência), estes limitados em 50 registros; (iv) Dados econômicos (índices como IPCA, IGP-M, Selic, etc.) e financeiros (despesas evitadas com aluguel e custos oportunidade do capital imobilizado); (v) Parâmetros da simulação de Monte Carlo (número de amostras), e; (vi) Parâmetros de importação e exportação (arquivo e formato).

Uma vez concluída a etapa de cadastro, o usuário deverá dar o comando ao sistema para que seja iniciada a Simulação de Monte Carlo. Dependendo do número de amostras informada pelo usuário, é possível prever que este cálculo possa levar alguns segundos ou até mesmo minutos. Ao final da simulação será apresentado ao usuário o valor esperado de desembolso para o intervalo de confiança de 95%. Este resultado é mostrado na seção de resultados, na parte superior direita da tela inicial.

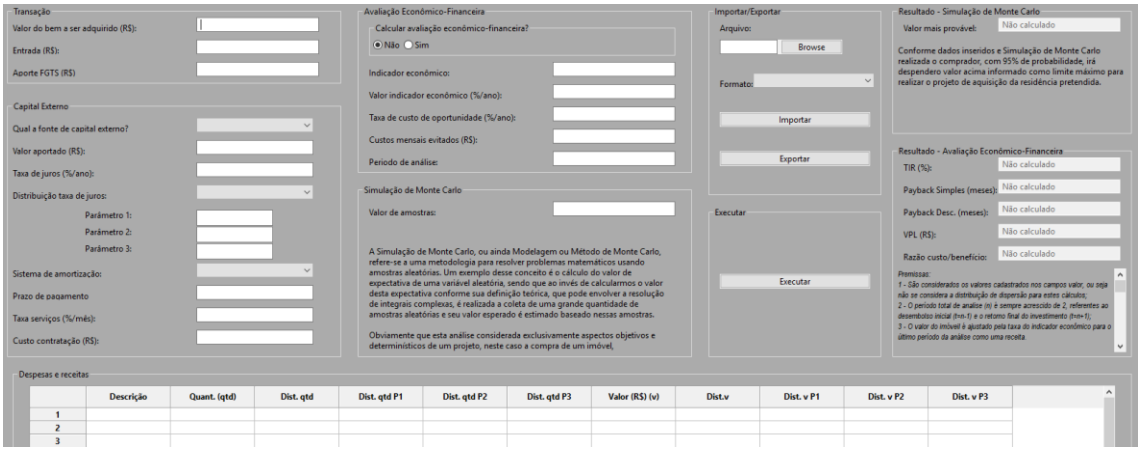

Figura 1 - Tela de cadastro

Caso o usuário tenha cadastrado dados econômicos e financeiros o sistema exibirá o resultado da análise de viabilidade econômica e financeira do projeto, sendo exibido ao usuário na seção de resultados, na parte inferior direita da tela inicial. Associado a cada um dos indicadores de viabilidade, é oferecido ao usuário uma breve descrição de como interpretar o respectivo indicador, conforme ilustrado na Figura 2. Os indicadores calculados são TIR – Taxa Interna de Retorno, VPL – Valor Presente Líquido, *Payback* Simples, *Payback* Descontado, e Razão Custo-Benefício. Por fim, o usuário possui a opção de salvar os dados de entrada num arquivo em formato **.csv* ou **.txt* para um eventual recalculo futuro do modelo, conforme ilustrado na Figura 3.

As fórmulas aplicadas nos cálculos do sistema estão disponíveis na Tabela 1.

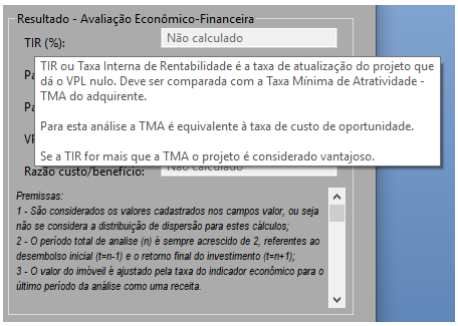

Figura 2 - Detalhe das explicações ao usuário

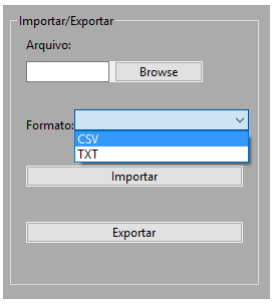

Figura 3 - Detalhe da importação e exportação de dados

Tabela 1 - Fórmulas matemáticas e financeiras

Descrição	Fórmula
Valores mínimos e máximos para intervalo de Confiança de 95% para Distribuição Normal [3]	$(\mu - 1.96 * \frac{\delta}{\sqrt{n}}\ e\ \mu + 1.96 * \frac{\delta}{\sqrt{n}}$ Sendo: μ: média; δ: desvio-padrão; n: quantidade de amostras.
TIR - Taxa Interna de Retorno [4]	$\sum_{i=1}^{n} \frac{FCi}{(1+TIR)^i} - FC0 = 0$ Sendo: FC: Fluxo de caixa; TIR: Taxa Interna de Retorno; n: quantidade de períodos.
VPL – Valor Presente Líquido [4]	$\sum_{i=1}^{n} \frac{FCi}{(1+TMA)^i} - FC0$ Sendo: FC: Fluxo de caixa; FC0: investimento inicial; TMA: Taxa Mínima de Atratividade. Para o projeto é considerado o valor cadastrado no campo taxa de custo de oportunidade; n: quantidade de períodos.
Payback Simples [5]	$\frac{Investimento\ inicial}{Fluxo\ de\ caixa\ do\ período}$
Payback Descontado [5]	$\frac{Investimento\ inicial}{Fluxo\ de\ caixa\ do\ período\ descontado}$ Para o cálculo do fluxo de caixa descontado, aplicar: $\sum_{i=1}^{n} \frac{FCi}{(1+TMA)^i}$ Sendo: FC: Fluxo de caixa; TMA: Taxa Mínima de Atratividade. Para o projeto é considerado o valor cadastrado no de taxa do indicador econômico.
Razão Custo-Benefício [5]	$\frac{\sum_{i=1}^{n} \frac{FCi}{(1+TMA)^i}}{Investimento\ inicial}$

DESENVOLVIMENTO DO *SOFTWARE* NA VERSÃO ORIENTADA A OBJETOS

A linguagem de programação foi C++ (pré-requisito da disciplina) empregando a IDE – *Integrated Development Environment* chamada *Code::Blocks* versão *17.12*, publicada sob *GNU - General Public License*, empregando como compilador *TDM-GCC MinGW* versão *5.1.0-2*, publicado sob GNU. Para a interface de usuário foi empregado a biblioteca *wxWidgets* versão *3.1.2*, publicada igualmente sob GNU. Além disso, o desenvolvimento foi apoiado pela ferramenta CASE – *Computer Aided Software Engineering* chamada *Astah UML* versão *8.1.0* comercializado pela empresa *Change Vision Inc*, empregado para a elaboração dos digramas UML – *Unified Modeling Language* . Para os testes foi empregado a ferramenta de planilhas eletrônicas *Excel* versão 2007, desenvolvido pela empresa *Microsoft*.

O sistema foi desenvolvido em 4 etapas, sendo elas: (i) levantamento de requisitos, (ii) análise e projeto, (iii) implementação ou codificação, e (iv) testes. Na primeira etapa do sistema, levantamentos de requisitos, foram identificados e analisados os principais requisitos funcionais do futuro sistema, conforme detalhado na Tabela 2. Destaca-se que a integralidade dos requisitos levantados nesta etapa foram analisados, implementados e testados.

Tabela 2 - Lista de requisitos funcionais

N°	Descrição	Peso atribuído
1	Permitir o cadastro de despesas de materiais, mão de obra, taxas, impostos, serviços, e outras despesas. O cadastro deverá permitir que o usuário escolha a distribuição de probabilidade dos valores e os principais parâmetros da distribuição (ex: tipo: material; descrição: porcelanato; quantidade: 100; distribuição_quantidade: mínimo, realista, máximo; custo unitário: 50; distribuição_custo: normal; parametro1: 5 (desvio padrão)). O cadastro deve poder ser lido a partir de um arquivo em pelo menos 2 formatos diferentes, p. ex. TXT, XML ou CSV	10 ptos
2	Permitir o cadastro de receitas, que podem ser oriundas de vendas que o usuário pretende realizar para viabilizar o projeto. O conceito de cadastro será similar ao de despesas. O cadastro deve poder ser lido a partir de um arquivo em pelo menos 2 formatos diferentes, p. ex. TXT, XML ou CSV	10 ptos
3	Permitir o cadastro de valor de venda do imóvel, valor de entrada e valor de aporte do FGTS. O cadastro deve poder ser lido a partir de um arquivo em pelo menos 2 formatos diferentes, p. ex. TXT, XML ou CSV	10 ptos
4	Permitir o cadastro de um capital externo, podendo ser um financiamento, empréstimo, hipoteca, etc. Deverá ser cadastrado o valor total a ser aportada, a modalidade de ressarcimento (ex: SAC, Price, etc.); taxa de juros, serviços contratados (ex: seguro), e despesas de cadastro. O cadastro deve poder ser lido a partir de um arquivo em pelo menos 2 formatos diferentes, p. ex. TXT ou XML.	10 ptos
5	Permitir o cadastro de despesas relativas a aluguel de imóvel. Esta entrada de dado será empregada na análise de viabilidade do projeto. O cadastro deve poder ser lido a partir de um arquivo em pelo menos 2 formatos diferentes, p. ex. TXT, XML ou CSV	10 ptos
6a	Permitir o cadastro de custos de oportunidade do capital que será imobilizado. Por exemplo, se atualmente o investidor possui estes recursos aplicados em Tesouro Direto e irá resgatar tais investimentos, esta referência pode ser empregada como custo de oportunidade pois este montante deixará de render nesta modalidade de investimento. Deverá prever vários tipos de modalidade de investimento, sendo que para cada uma deverá ser cadastrado: descrição (ex: poupança, tesouro direto, etc.), taxa de oportunidade (ex: 6,5% ao ano, 0,75% ao mês, etc.). Esta entrada de dado será empregada na análise de viabilidade do projeto. O cadastro deve poder ser lido a partir de um arquivo em pelo menos 2 formatos diferentes, p. ex. TXT, XML ou CSV	10 ptos
6b	Permitir o cadastro de custos evitados, como por exemplo aluguel da atual residência. O cadastro deve poder ser lido a partir de um arquivo em pelo menos 2 formatos diferentes, p. ex. TXT, XML ou CSV	-
7	Permitir o cadastro de indicadores econômicos para o cálculo da análise de viabilidade. Permitir o cadastro dos seguintes indicadores: IPCA, IGP-M e Selic. Esta entrada de dado será empregada na análise de viabilidade do projeto. O cadastro deve poder ser lido a partir de um arquivo em pelo menos 2 formatos diferentes, p. ex. TXT, XML ou CSV	10 ptos
8	Permitir que a Simulação de Monte Carlo seja iterativa, permitindo que o usuário ajuste os parâmetros de entrada a cada nova simulação. Ou seja, os dados de entrada não devem ser apagados ao final da simulação, de forma a permitir uma alteração ágil dos dados de entrada do modelo.	0 ptos
9	Permitir que seja importado e exportado em formato de arquivo CSV os dados de entrada do modelo, permitindo que o usuário salve e carregue tais dados de forma rápida.	0 ptos
10	Realizar uma Simulação de Monte Carlo para determinar o valor mais provável de aquisição da residência com base num intervalo de confiança de 95%. A simulação deverá considerar os parâmetros oriundos dos requisitos n°1, 2, 3 e 4. O usuário poderá cadastrar o número de amostras desejado, sendo o cadastro padrão 10.000 (dez mil) amostras.	10 ptos
11	Realizar uma análise de viabilidade econômica e financeira. A análise deverá considerar os parâmetros	10 ptos

N°	Descrição	Peso atribuído
	cadastrados nos requisitos n°1, 2, 3, 4, 5 e 6. Os índices a serem calculados deverão ser: Taxa Interna de Retorno (TIR), Valor Presente Líquido (VPL), *payback* simples e descontado, Razão Benefício-Custo. Para o cálculo deverão ser adotadas premissas de projeção futura com base nos indicadores econômicos cadastrados no requisito n°7.	
12	Gravar os resultados da análise de viabilidade financeira em pelo menos 2 formatos diferentes, p. ex. TXT, XML ou CSV.	10 ptos

Na etapa seguinte, análise e projeto, os requisitos identificados na etapa anterior foram modelados visando obter os principais diagramas de classe, diagramas de sequencia e digramas e atividades. O diagrama de classes resumido pode ser observado na Figura 4. Para uma visão detalhada do projeto de *software* consultar o Anexo I.

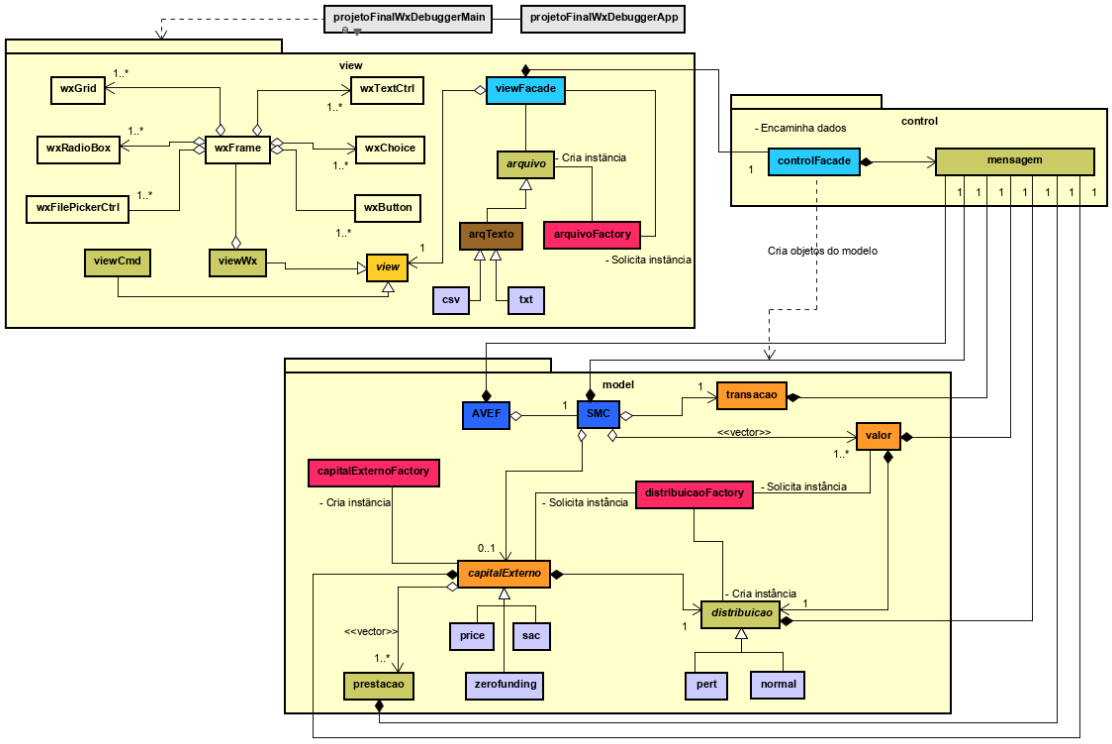

Figura 4 - Diagrama de classes (resumido)

Foi empregado um Padrão de Projeto [6] tipo MVC - *Model View Controller* para as camadas de interface de usuário (pacote *view*), controlador (pacote *control*) e modelo de negócio (pacote *model*). Para o pacote v*iew* foram empregados os seguintes Padrões de Projeto: *Façade* para a comunicação com o pacote *control* (classes *viewFacade* e *controlFacade*); DAO – *Data Access Objet* para a persistência de dados (vide classes *arquivo*, *arqTexto*, *csv* e *txt*); *Abstract Factory* para a criação de objetos da classe *arquivo* (vide classe *arquivoFactory*); *Bridge* para desacoplar a implementação da interface de usuário do pacote *View* (vide classes *viewFacade*, *view*, *viewCmd* e *viewWx*). No pacote *control* foi empregado o uso do Padrão de Projeto *Façade* para simplificar a comunicação com os demais pacotes. Este pacote implementa igualmente a classe *mensagem* que trata de gerenciar a comunicação entre as diferentes camadas do sistema, trazendo consigo um indicador de erro e eventuais registros de texto com as explicações do erro observado. No pacote *model* foi empregado o Padrão de Projeto *Abstract Factory* para a criação de objetos das classes *capitalExterno* e *distribuicao* (vide classes *capitalExternoFactory* e *distribuicaoFactory*). Cabe destacar que, considerando que as implementações do Padrão de Projeto *Façade* são instâncias únicas no sistema (*viewFacade* e *controlFacade*), o Padrão de Projeto *Singleton* acabou sendo indiretamente implementado.

Com relação ao pacote *view* foi necessário realizar a implementação de duas interfaces gráficas distintas, uma vez que uso da biblioteca *wxWidgets* não permitia a depuração do código (*debugging*). Na verdade esta funcionalidade, por razões desconhecidas ao autor, oscilava entre um funcionamento normal ao esperado de qualquer IDE, com parada em todos *breakpoints* previamente selecionados, ao total desprezo a tais marcações,

executando o código como se estivesse fora do modo de depuração. Portanto, para não prejudicar a qualidade do *software* e agilizar o processo de desenvolvimento, fez-se necessário a criação de uma interface gráfica em linha de comando, através da classe *viewCmd*.

Especificamente sobre o pacote *model, e*ste opera com 2 classes principais, chamadas *AVEF* e *SMC*, que caracterizam as principais funções do sistema, o cálculo de uma avaliação econômico-financeira, e daí o acrônimo *AVEF*, e a Simulação de Monte Carlo propriamente, daí a sigla *SMC*. A classe *SMC* agrega 3 classes importantes para o sistema, sendo elas: *transação* (com os dados base da aquisição do imóvel), *capitalExterno* (com os dados relativos ao financiamento, hipoteca, empréstimo, etc. como taxa de juros e sua respectiva variação, sendo este agrega a classe *prestação* através de uma lista tipo *vector*) e *valor* (com os registros de despesas e receitas e suas variações, sendo esta agregada através de uma lista tipo *vector)*. Tanto a classe *capitalExterno* quanto *valor* possuem uma agregação com a classe *distribuição* e esta, nas suas diferentes implementações (vide classes *normal* e *uniforme),* possui método específico para a geração de valores aleatórios, característica fundamental do Método de Monte Carlo. Destaca-se aqui a limitação da implementação adotada, empregando a classe *default_random_engine* , pois a geração de números é pseudorrandômica [7].

Conforme já abordado, a Simulação de Monte Carlo ocorre ao realizar uma massiva variação aleatória de valores visando determinar o valor máximo mais provável para dado fenômeno em estudo. Neste projeto a simulação ocorre propriamente dito na classe *SMS* através do método *calcularSmc()*. Segue extrato de código deste método, onde o sistema realiza as iterações da simulação.

```
1. //Execução de amostragem na quantidade de amostras definida pelo usuário.
2. double vt = 0; // Valor total.
3. vector<double> resultados; // Vetor que acumula os múltiplos resultados.
4. double vt1 = getTransacao()->getEntrada() + getTransacao()->getAporteFgts();//Valores iniciais correspondem às
   entradas básicas da transação.
5. for (int i = 0; i<=getAmostras(); i++){
6.    //Reiniciar vt.
7.    vt = vt1;
8.    //Amostra e cálculo para Capital Externo.
9.    getCapitalExt()->calcularPrestacoes(getCapitalExt()->getDist()->gerarAleatorio());
10.      vt += getCapitalExt()->getValorAmortizacao() + getCapitalExt()->getValorJuros() + getCapitalExt()-
   >getValorServicos() + getCapitalExt()->getCustoContratacao();
11.      //Amostra e cálculo para os Valores.
12.      double v1, v2;
13.      for (int j = 0; j<getValores().size(); j++){
14.         v1 = getValores()[j]->getDistQ()->gerarAleatorio();
15.         v2 = getValores()[j]->getDistU()->gerarAleatorio();
16.         vt += v1 * v2;
17.      }
18.      resultados.push_back(vt);
19.   }
```

Na linha 2 é iniciada a variável que acumula os diversos valores do projeto para aquela iteração, sendo iniciado em zero e na linha 4 o sistema soma valores fixo iniciais do projeto (valor de entrada e aporte de FGTS). A partir da linha 5 é iniciada a amostragem, sendo que na linha 7 o valor da variável com os valores fixos é atribuído a um parâmetro que realizará a soma dos valores variáveis para aquela iteração. Em seguida na linha 9 sistema recalcula os valores do capital externo (juros, amortização e serviços) com base numa amostra aleatório da taxa de juros e na linha 10 os subtotais são acrescidos à variável de valor total da iteração. A partir da linha 12 é iniciada a leitura e soma dos valores referentes aos valores de despesas e receitas, sendo que é obtida uma amostra aleatória de quantidade e preço unitário para cada registro de valor e os subtotais acrescidos ao total daquela iteração (linhas 14 a 16). Ao final, o valor total para aquela amostra é inserido no vetor de resultados (linha 18). Com base nesse vetor, de tamanho igual ao total de amostras, são calculados os parâmetros da distribuição normal que representa a dispersão dos valores do total de investimentos para a aquisição da propriedade. E, finalmente, com base nos parâmetros descritivos desta curva normal, o sistema calcula o valor referente ao intervalo de confiança de 95%.

Para a implementação da classe *AVEF* foi necessário o reuso de código externo ao projeto, devido a dificuldade técnica em implementar um método para o cálculo da fórmula para obter a TIR do projeto. Para tanto, foi empregado o código disponível em [8]. Cabe ressaltar que para esta etapa, cálculo da Avaliação Econômico-Financeira, não foram usados os valores de distribuição cadastrados pelo usuário. Os valores empregados para efeitos de cálculos foram aqueles registrados nos campos nominais (ex: taxa de juros, valor unitário, quantidade).

Destaca-se que as fases de codificação e testes foram conduzidas em paralelo para cada pedaço do sistema, sendo que testes unitários eram realizados na medida cada classe e/ou método era concluído. Ao final,

com o sistema integralmente finalizado, foram realizados novos testes para validar o conjunto de cálculo das regras de negócio. Para esta fase foi utilizado como ferramenta e comparação o aplicativo *Excel* visando verificar se os cálculos financeiros estavam aderentes ao resultado esperado e calculado pela planilha eletrônica.

O código do *software* objeto deste trabalho está disponível para consulta no Anexo II.

TABELA DE CONCEITOS UTILIZADOS E NÃO UTILIZADOS

Segue na Tabela 3 resumo dos principais conceitos empregados e não empregados ao longo do desenvolvimento deste projeto de software.

Tabela 3 – Lista de conceitos utilizados e não utilizados no trabalho

N°	Conceito	Uso	Onde	Situação/Justificativa
1	**Elementares:**			
	Classes, objetos	Sim	Todos .h e .cpp.	Utilizados para implementar conceitos básicos de OO.
	Atributos (privados), variáveis e constantes	Sim	Todos .h e .cpp.	Utilizados para armazenar valores básicos da modelagem matemática.
	Métodos (com e sem retorno)	Sim	Todos .h e .cpp.	Utilizados para processar valores básicos da modelagem matemática.
	Métodos (com retorno const e parâmetro const).	Não	-	Não foi identificada a necessidade.
	Construtores (sem/com parâmetros) e destrutores	Sim	Todos .h e .cpp.	Utilizados para implementar conceitos básicos de OO.
	Classe Principal	Sim	*projetoFinalWxDebuggerApp* e *projetoFinalWxDebuggeMain.*	Utilizados para implementar as classes que chamam os métodos principais do sistema.
	Divisão em .h e .cpp	Sim	No projeto.	Utilizados para implementar conceitos básico de OO em linguagem C++.
2	**Relações de:**			
	Associação	Sim	*capitalExterno* e *capitalExternoFactory; distribuicao* e *distribuicaoFactory; arquivo e arquivoFactory, viewFacade* e *arquivo.*	Relacionamento entre as classes alvo e as *Abstract* F*actory.* Relacionamento entre a classe *Façade* do pacote *View* para persistir os dados. A simples associação garante o fluxo de dados.
	Agregação via associação	Sim	*viewWx* e todos os elementos gráficos; *viewFacade* e *view; SMC, valor, transação* e *capitalExterno; capitalExterno* e *prestação.*	Foi utilizada a agregação, pois estes elementos são independentes, contudo, agregados para compor um novo objeto com novas funcionalidades.
	Agregação propriamente dita	Sim	Todas as classes com a classe mensagem (*controlFacade, AVEF, SMC, transação, distribuicao, prestação, capitalExterno, valor); viewFacade* e *controlFacade.*	A agregação forte com as classes que utilizam da classe mensagem foi feita como agregação forte pois os objetos inexistem sem que um objeto mensagem tenha sido instanciado. Para o relacionamento entre as *Façades* o raciocínio é semelhante, ou seja, um objeto da classe *viewFacade* não pode existir sem que haja um objeto *controlFacade* em sua composição. Essa premissa garante a existência nativamente da comunicação entre as duas camadas do sistema.
	Herança elementar	Sim	*arquivo e arqTexto; arqTexto, csv e txt; view, viewWx e viewCmd; capitalExterno, sac, price e zeroFunding; distribuição, uniforme* e *normal.*	Estas classes representam especializações dentro do modelo, sendo que desta forma parte do código da super classe é reaproveitado na sub classe.
	Herança em diversos níveis	Sim	*arquivo, arqTexto, csv* e *txt.*	Foi realizado desta forma pois as duas formas de persistência de dados são bastante similares,

N°	Conceito	Uso	Onde	Situação/Justificativa
				distinguindo tão somente pelo tipo de separador utilizado. Assim, elas foram especializadas alterando apenas este atributo, empregando os mesmos métodos das super classes, tanto *arqTexto* quanto *arquivo*.
	Herança múltipla	Não	-	Não foi identificada a necessidade.
3	**Ponteiros, generalizações e exceções:**			
	Operador *this*	Sim	*projetoFinalWxDebuggerMain*.	Para relacionar os elementos gráficos com a janela (*Frame*) que eles se relacionam.
	Alocação de memória (new & delete)	Sim	p*rojetoFinalWxDebuggerMain, controlFacade, AVEF, SMC, capitalExterno, capitalExternoFactory, distribuição, distribuicaoFactory, prestação, price, sac, transação, valor, zeroFunding, arquivo* e *viewFacade*.	Alocação de memória para os diferentes tipos de relacionamentos (ex: agregação).
	Gabaritos/Templates criada/adaptados pelos autores (e.g. Listas Encadeadas via Templates)	Não	-	Falta de tempo para análise e implementação de solução.
	Uso de Tratamento de Exceções	Não	-	Não foi identificada a necessidade.
4	**Sobrecarga de:**			
	Construtoras e Métodos	Sim	Pacote m*odel*.	Utilizados para implementar conceitos básicos de OO. Empregado para simplificar a instanciação quando novos parâmetros são acrescentados à construtora.
	Operadores	Não	-	Não foi identificada a necessidade.
	Persistência de Objetos:			
	Texto via Arquivos de Fluxo	Sim	*arquivo, arqTexto, csv* e *txt*.	Escrita e leitura de arquivos tipo CSV e TXT, conforme requisitos do sistema.
	Binário	Não	-	Não foi identificada a necessidade.
5	**Virtualidade:**			
	Métodos Virtuais	Sim	*capitalExterno, distribuicao, arquivo* e *view*.	Uso necessário para remeter a execução de determinados métodos para as classes especializadas.
	Polimorfismo	Sim	*capitalExterno, price e sac; distribuicao, uniforme e normal; view, viewWx* e *viewCmd*.	Para capital externo o método de calcular as prestações é diferente para as duas sub classes. Para a distribuição o método para gerar números aleatórios depende de cada tipo distribuição de probabilidade. Para a interface de usuário (classe view) os métodos de ler e exibir dados são diferentes para cada tipo de biblioteca gráfica empregada.
	Métodos Virtuais Puros / Classes Abstratas	Não	-	Não foi identificada a necessidade.
	Coesão e Desacoplamento	Sim	No projeto.	Conceitos básicos de OO, utilizados ao longo de toda a análise e codificação.
6	**Engenharia de Software:**			
	Levantamento de Requisitos Textualmente e Tabelado	Sim	No projeto.	Vide Tabela 2.
	Diagrama de Classes em UML	Sim	No projeto.	Vide Figura 4 e Anexo I.
	Outros diagramas em UML	Sim	Pacote *model*.	Diag. de Casos de Uso, Diag. de Atividades, Diag. de Seqüência, Diag. de Pacotes.

N°	Conceito	Uso	Onde	Situação/Justificativa
				Vide Anexo I.
7	**Biblioteca Gráfica:**			
	Funcionalidades Elementares	Sim	*viewWx*	Emprego da biblioteca *wxWidgets* para a interface com o usuário.
	Funcionalidades Avançadas	Não	-	Não foi identificada a necessidade.
	Interdisciplinaridades por meio da utilização de Conceitos de Matemática, Física etc			
	Ensino Médio (especificar quais Conceitos aqui)	Sim	*price*, *sac*, *SMC* e *AVEF*.	Emprego de operadores aritméticos básicos (soma, subtração, multiplicação e divisão) bem como raiz quadrada.
	Ensino Superior (especificar quais Conceitos aqui)	Sim	*SMC* e *AVEF*.	Emprego de conceitos ciências econômicas e matemática financeira para o cálculo de TIR, VPL, *Payback* Simples e Descontado e Razão Custo-Benefício. Emprego de conceitos de estatística descritiva para o cálculo das distribuições de probabilidades.
8	**Organizadores:**			
	Espaço de Nomes (Namespace) ou pacotes criados pelos autores	Sim	Pacotes *model*, *view* e *control.*	Melhor organização do código através de chamadas pelo *namespace*.
	Classes aninhadas	Não	-	Não foi identificada a necessidade.
	Estáticos e String:			
	Atributos estáticos e chamadas estáticas de métodos	Não	-	Falta de tempo para análise e implementação de solução e inexperiência no uso de métodos estáticos em C++.
	A classe Pré-definida String ou equivalente	Sim	No projeto.	Uma vez que não existem requisitos não funcionais ou restrições de projeto para o uso de memória, deu-se preferência para o uso da classe *String* pelas vantagens no processamento de texto nativo daquela classe.
9	**Standard Template Library (STL):**			
	Vector da STL (p/ objetos ou ponteiros de objetos de classes definidos pelos autores)	Sim	*controlFacade*, *mensagem*, *AVEF*, *SMC*, *capitalExterno*, *distribuicao*, *price*, *sac*, *uniforme*, *normal*, *transação*, *valor*, *zeroFunding*, *arqTexto*, *csv*, *txt*, *view*, *viewFacade.*	Todas as listas do projeto foram implementadas empregando *vector* da STL. Uma vez que não existem requisitos não funcionais ou restrições de projeto para o uso de memória, deu-se preferência para o uso de *vector* pelas vantagens no acesso e incremento de elementos nativo daquela classe.
	List da STL (p/ objetos ou ponteiros de objetos de classes definidos pelos autores)	Não	-	Não foi identificada a necessidade.
	Pilhas, Filas, Bifilas, Filas de Prioridade, Conjuntos, Multi-Conjuntos, Mapas ou Multi-Mapas	Não	-	Não foi identificada a necessidade.
10	**Uso de Conceito Avançado no tocante a Orientação a Objetos:**			
	Padrões de Projeto: GOF	Sim	No projeto (para maiores detalhes consultar seção anterior).	MVC - *Model View Controller*, *Façade;* DAO – *Data Access Objet*, *Abstract Factory*; *Bridge* e *Singleton*.
	Programação orientada a eventos e visual:Objetos gráficos como formulários, botões etc (Listar apenas os utilizados)	Sim	*viewWx*	Emprego da biblioteca *wxWidgets* para a interface com o usuário.
	Programação concorrente:Threads (Linhas de Execução) no âmbito da Orientação a Objetos,	Não	-	Recomendado do uso de *threads* como melhoria futura.

N°	Conceito	Uso	Onde	Situação/Justificativa
	utilizando Posix, C-Run-Time, Win32API ou afins (com ou sem uso de Mutex, Semáforos, ou Troca de mensagens)			
	API de Comunicação em Rede: Cliente Servidor	Não	-	Não foi identificada a necessidade.

DISCUSSÃO E CONCLUSÕES

Considerando que todos os requisitos inicialmente levantados foram devidamente analisados, codificados e testados, considera-se que a peça de *software* entregue é plenamente funcional conforme projeto inicial. O produto entregue pode ser considerado de média complexidade, e aderente ao objetivo proposto para a disciplina de Programação Avançada. A linguagem C++ é exigente com variações de sintaxe e semântica muito sutis, difíceis de serem identificados para novos adeptos, e que quando não percebidos podem levar a comportamentos indesejados na sua execução. Contudo, a maior curva de aprendizado se deu na combinação de uso da biblioteca da interface gráfica *wxWidgets*, IDE *Code Blocks* e compilador *TDM-GCC MinGW* uma vez que a a configuração do ambiente foi delicada e inconstante. Delicada pela necessidade de parâmetros muito específicos e não muito intuitivos para que o funcionamento esperado fosse atendido; e inconstante, pois para o caso de depuração (*debuggin*g) não foi possível obter o desempenho esperado. A depuração ocorria por alguns ciclos, todavia, por razões desconhecidas, a partir de dado momento os *breakpoints* eram sumariamente ignorados pelo compilador. Por esta razão o autor adotou uma solução de contorno implementando um interface em linha de comando exclusiva para a execução de *debugging* e testes.

O *software* desenvolvido possui potencial para o emprego de *multi-threads*, pois conforme demonstrado no Gráfico 1 o tempo de execução é diretamente proporcional ao número de amostras requeridas pelo usuário. O uso de múltiplas *threads* paralelas, aproveitando o máximo de equipamentos dotados de vários núcleos de processamento, permitiria a redução dos tempos demonstrados no gráfico. Obviamente que deve se considerar que o modelo matemático não é de alta complexidade, contudo o cálculo massivo de amostras consome recursos significativos de processamento (em torno de 17% com 10 milhões de amostras).

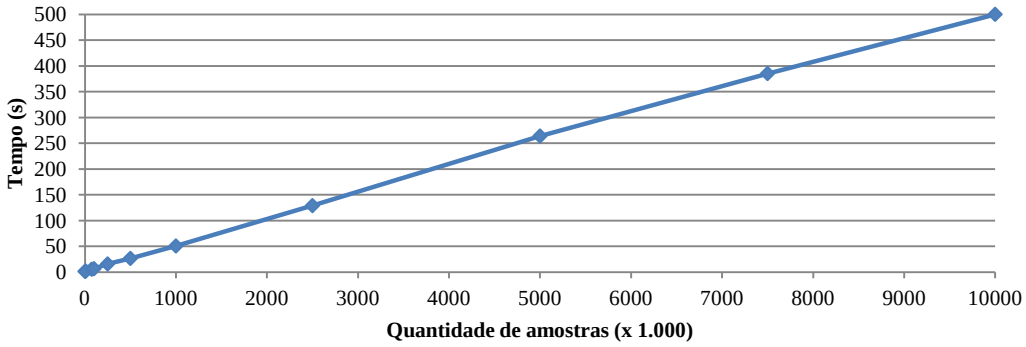

Gráfico 1 - Relação entre quantidade de amostras e tempo de processamento

Quando se fala de tópicos de Engenharia de *Software* é impossível desviar do assunto custo, pois, todo projeto é composto, fundamentalmente, pelo tripé escopo, prazo e custo. O escopo foi abordado ao longo deste trabalho, e o elemento prazo foi definido como sendo o período entre os dias 07/03/2019 e 20/05/2019 (início e fim do quadrimestre letivo), ou seja, um total de 74 dias corridos. Quanto à dimensão custo, o autor buscou registrar as horas investidas para a elaboração do trabalho a fim de estimar o total de homens/hora necessários para um projeto deste porte. Os dados obtidos indicam um total de 88 horas, compreendendo as etapas de levantamento de requisitos, análise e elaboração de diagramas (23%), elaboração da Interface Gráfica com a biblioteca *wxWidgets* (18%) , codificação e testes (59%).

A respeito da solução obtida, cabe nesta seção abordar os potenciais e limitações do projeto. Com relação ao potencial, conforme já abordado, todas as regras de negócio relativas à aquisição de uma residência são semelhantes a projetos de maior envergadura. Algumas diferenças seriam múltiplas fontes de capital externo, um número muito maior de despesas e receitas, tratamento de riscos, dentre outros. Todavia, considerando que a abordagem do projeto é uma simplificação de algo maior, seria possível, a partir deste código, explorar a escalabilidade do modelo proposto visando atender necessidades de projetos de maior porte, como infraestrutura, varejo, etc.

Sobre as limitações destaca-se principalmente o número de linhas na área dedicada ao cadastro de valores de despesa e receitas, neste caso limitado a 50. Esta decisão de projeto foi tomada visando minimizar o tempo de desenvolvimento com questões de interface de usuário. Obviamente que existem diversas soluções para esta restrição, algumas mais simples outras mais elaboradas, contudo, todas elas iriam apropriar ainda mais horas de codificação, aumentando o custo do projeto sem acrescentar real valor ao produto final. Isso por que a quantidade de despesas e receitas associadas ao escopo selecionado em raras situações ultrapassaria o limite definido para o projeto (50 registros). Ainda sobre as limitações cabe destacar que uma das principais características da Simulação de Monte Carlo é a variação aleatória de parâmetros. Contudo, conforme brevemente já abordado, o método empregado pelo autor é pseudorrandômico. Mesmo sendo esta uma limitação identificada, é imaturo avaliar eventuais impactos nos resultados devido a esta natureza pseudorandômica. A avaliação deste efeito caberia a um trabalho específico sobre o tema, sendo que cabe ao autor destacar esta característica do produto de *software* entregue.

Ademais, existem algumas melhorias que poderiam ser abordadas numa versão futura do *software* entregue. A primeira delas é uma melhoria de funcionalidade, visando integrar a Simulação de Monte Carlo com a Avaliação Econômico-Financeira. O escopo deste trabalho previu calcular exclusivamente o valor final de aquisição e reforma da residência, ou seja, não era um requisito original ao longo da simulação registrar e calcular os valores intermediários para as despesas e receitas cadastradas (classe *valor*) e capital externo (classe *capitalExterno*). Estes valores serviriam apenas como entrada para calcular a distribuição de valores do preço final. Todavia, quando da implementação da Avaliação Econômico-Financeira percebeu-se que se os valores simulados para cada iteração tivessem sido registrados, de forma individualizada e conforme sua respectiva distribuição de quantidade e valor unitário isto resultaria numa evolução da solução inicialmente proposta, pois teríamos valores de entrada de maior confiabilidade para o cálculo da atratividade da aquisição e reforma. Portanto, o autor recomenda, para uma futura versão do sistema, esta integração de forma a obter melhores resultados na etapa de Avaliação Econômico-Financeira. Existem ainda duas melhorias possíveis, mas estas de cunho de requisitos não-funcionais, e que melhorariam a leitura do código fonte por entes externos ao projeto. A primeira melhoria seria o emprego de *templates*, principalmente na classe *view* quando é realizada a leitura dos valores da Interface Gráfica. Boa parte das operações ali realizadas são repetitivas e iguais para vários parâmetros. Portanto, poderia ser empregado um *template* ou até mesmo uma classe estática para simplificar o código. Outra melhoria de natureza semelhante é sobre o uso do Padrão de Projeto *Iterator* para percorrer as listas de valores do sistema. Esta melhoria traria uma maior desacoplamento do modelo em relação ao tipo de lista escolhida para a implementação de relação de agregação entre os objetos. O autor não optou por estas implementações à época da codificação devido a pouca experiência na manipulação destes recursos em linguagem C++ (*templates* e classes estáticas) ou mesmo desconhecimento da funcionalidade (*Iterator*).

REFERÊNCIAS

[1] AMELIN, M. *Monte Carlo Simulation in Engineering*. Acessado em 11 de março de 2019 às 21h55: https://www.kth.se/social/files/562dfd19f276544062a47ba1/Monte%20Carlo%20Simulation%20in%20Engineering.pdf.

[2] ATOMIC HERITAGE FOUNDATION. *Computing and the Manhattan Project.* Acessado em 11 de março de 2019 às 22h03: https://www.atomicheritage.org/history/computing-and-manhattan-project

[3] DA FONSECA, J. S., DE ANDRADE MARTINS, G. *Curso de Estatística*. 3ª Edição. Atlas. 1981.

[4] BARBIERI, J. C., ÁLVARES, A. C. T., MACHLINE, C. *Taxa Interna de Retorno: controvérsias e interpretações.* Acessado em 12 de maio de 2019 às 20:26: https://pesquisa-eaesp.fgv.br/sites/gvpesquisa.fgv.br/files/arquivos/alvares_-_taxa_interna_de_retorno_controversias_e_interpretacoes.pdf

[5] HUMMEL, P. R. V., PILÃO, Nivaldo Elias. *Matemática Financeira e Engenharia Econômica: Teoria e Prática da Análise de Projetos de Investimento*. São Paulo. Thomson. 2004

[6] GAMMA, E., HELM, R., JOHNSON, R., VLISSIDES, J. Padrões de Projeto: Soluções reutilizáveis de software orientado a objetos. Porto Alegre. 2007. Bookman.

[7] CPLUSPLUS.COM. Acessado em 11 de maio de 2019 às 19h11: http://www.cplusplus.com/reference/random/default_random_engine/

[8] CODE PROJECT. *Internal Rate of Return (IRR) Calculation*. Acessado em 11 de maio de 2019 às 19h23: https://www.codeproject.com/Tips/461049/Internal-Rate-of-Return-IRR-Calculation