2
Fundamentos Teóricos

2.1
Visualização de Informação

Visualização de informação é uma ramificação das áreas de computação gráfica e de interface com o usuário. Ela é centrada na forma como os dados são apresentados para o usuário visando um bom entendimento do conteúdo. A computação gráfica iniciou-se sendo utilizada para estudar problemas científicos. Entretanto a falta de tecnologia limitou em muito o seu uso. Com o passar do tempo e o avanço tecnológico a computação gráfica abriu as portas para desenvolvimento da área de visualização de informação criando novas possibilidades e acima de tudo aumentando o interesse para essa área de pesquisa.

Um outro incentivador foi o também desenvolvimento da área de interface com usuário. Equipamentos de geração de imagem, som, computadores mais eficientes, transmissão de dados mais veloz, novos dispositivos físicos de interação, etc.

Então, combinando aspectos de computação gráfica, interfaces homem-computador e mineração de dados, a visualização de informações permite a apresentação de dados em formas gráficas de modo que o usuário possa utilizar sua percepção visual para melhor analisar e compreender as informações [2].

Em geral, a visualização de informações [2, 3] permite ao usuário obter uma representação visual que, por um lado abstrai detalhes do conjunto de informações, mas por outro propicia uma organização desse conjunto segundo algum critério.

Desta forma a representação de certo domínio de aplicação deve utilizar técnicas de visualização de informação para permitir ao homem perceba, interprete e compreenda os dados apresentados e assim deduza novos conhecimentos.

Diferente da visualização científica, que representa dados com geometria conhecida, visualização de informação tem que criar metáforas visuais para representar dados abstratos que não possuem uma geometria natural. Para isso é necessário criar formas de visualização que permitam a interpretação e compreensão além de possibilitar mecanismos de interação e manipulação.

Desenvolvedores devem considerar a melhor forma de mapear a informação para uma interface gráfica que facilite a sua interpretação para os seus usuários, como fornecer meios que permitam limitar a quantidade de informações que estes recebem, mantendo-os, ao mesmo tempo, "cientes" do espaço total de informação. É também necessário possibilitar formas de manipulação do conjunto de dados, tanto geométrica (rotações e zoom na representação gráfica, por exemplo) como analiticamente (redução ou expansão do conjunto de dados exibido de acordo com algum critério determinado pelo usuário) [4].

Existem diversas alternativas visuais para visualização de informação, mas o princípio básico da navegação e busca de informação pode ser resumido através do "mantra" da busca de informação visual: Overview first, zoom and filter, then details-on-demand [5]. Um sistema que implementa esta técnica permite ao usuário alterar o seu foco de visão sem perder o contexto, filtrar e agrupar dados relacionados e consultar para recuperar informações mais detalhadas.

Seguindo esta consideração, com um sistema que permita uma boa visualização de informação, interfaces de manipulação direta e uma possibilidade de consulta dinâmica, o usuário poderá assumir tarefas muito mais ambiciosas [5].

É com este intuito que framework aqui desenvolvido busca atender todos estes conceitos. Dentre suas contribuições ele possibilita ao desenvolvedor:

- Mapear a informação da maneira mais adequada à interpretação e compreensão do seu usuário (desenvolvedor deve conhecer o perfil do seu usuário final);
- Manipulação direta dos dados através da interface com a possibilidade de alteração da semântica utilizada nas interações;
- Atribuir e selecionar dados particulares da informação que poderão disponibilizados para a consulta do usuário.

2.2
Facetas

Ranganathan foi o primeiro a introduzir ao mundo a palavra “faceta” na ciência da computação e na biblioteconomia e também foi o primeiro a desenvolver a teoria de análise de facetas [6]. Originalmente facetas foram definidas como aspectos, propriedades ou características de uma classe ou tópico específico, definidas claramente, e deviam ser mutuamente exclusivas e exaustivas.

Recentemente diversos autores têm feito menção a facetas, classificação facetada e análise de facetas com diferentes conotações. Análise de facetas é definida como a organização de termos de um domínio de conhecimento em facetas homogêneas, mutuamente exclusivas [6].

Cada faceta é derivada de um “ancestral” através de uma única característica de divisão. Cada categoria lógica deve ser isolada e cada nova característica de divisão deve ser indicada claramente. Portanto, uma faceta consiste em um grupo de termos que representa uma e somente uma característica de divisão de um tópico pertencente a um domínio.

Resumindo, a idéia é dividir um tópico em facetas de alto nível ou categorias fundamentais. Cada faceta de alto nível é subdividida em componentes, as sub-facetas, que representam diferentes características.

2.2.1
Classificação Facetada

A classificação facetada permite que um mesmo objeto possua diversas classificações, permitindo assim a busca e a navegação por classes que formam grupos de objetos. Ela difere de uma classificação tradicional, pois a mesma não atribui entalhes fixos aos assuntos em seqüência, mas utiliza uma definição clara, mútua exclusividade, e aspectos coletivos, propriedades ou características de uma classe ou assunto específico. Tais aspectos, propriedades, ou características são chamados de facetas de uma classe ou de um assunto, um termo introduzido na teoria da classificação e dado este significado novo pelo bibliotecário e pelo classificador indiano S.R. Ranganathan e usado primeiramente em sua classificação (Collon Classification) no início dos anos 30 [7].

Um dos benefícios principais da classificação facetada é que mesmo se você não souber o nome de um objeto, você pode conseguir um entendimento compartilhado bem preciso do que é, descrevendo em termos de diversas, e mútuas categorias de informação. Se estiver tentando o descrever um refrigerador, por o exemplo, pode-se chegar ao que é, com precisão, especificando seu tamanho, a substância que é feito, a sua cor, sua posição típica em uma casa, e sua função.

No ambiente computacional, o conjunto de facetas não necessita ser fechado, durante a classificação ou a recuperação da informação por buscadores. Os criadores de uma base de dados compartilhada podem adicionar uma nova faceta a qualquer hora, e os usuários podem selecionar elementos de quantas facetas quiserem.

Unidades de informação podem ser associadas com elementos de diferentes hierarquias de facetas. Desta forma, buscadores de informação podem passar por diversas hierarquias para ver quais informações estão associadas com determinado elemento. Novas hierarquias podem ser adicionadas a qualquer momento. Elas simplesmente representam uma perspectiva de interesse de pelo menos um espectador.

A navegação facetada permite que usuários naveguem pela informação selecionando progressivamente as facetas dos itens de informação. Em uma navegação facetada os assuntos são divididos em facetas (aspectos), e o número de classes são resultados da classificação.

Inicialmente o usuário escolhe uma das facetas dentre o conjunto de facetas disponível. Posteriormente o usuário poderá escolher outras facetas sucessivamente dentre o conjunto de facetas disponível após a filtragem executada pelas facetas por ele escolhidas anteriormente. O conjunto de facetas que o usuário terá disponível para navegar é dependente do caminho por ele escolhido.

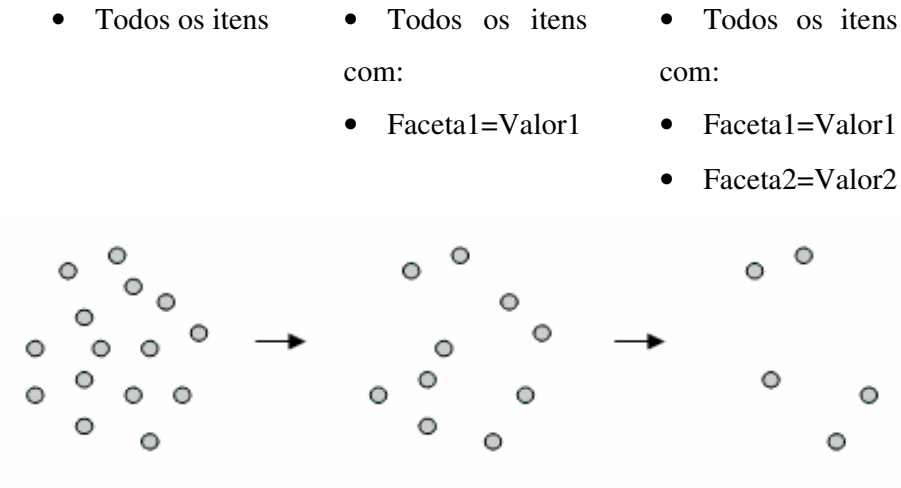

Figura 1 – Seleção de Facetas

2.3 Manipulação Direta

Para que os usuários aprendam e interajam facilmente com sistemas de computador, eles devem compreender a progressão e a ação de cada etapa de suas ações. Os estudos de manipulação direta descrevem os sistemas interativos onde o usuário interage fisicamente com seu sistema. A característica fundamental de tal sistema é o controle do usuário. Ao invés de digitar comandos e de permitir que o sistema aja como um intermediário, um sistema de manipulação direta permite que o usuário sinta como se ele estivesse no controle, permitindo que interaja fisicamente com a informação exibida, e apresentando uma representação visual do progresso da ação e de sua conclusão.

A noção de "manipulação direta" não é um conceito unitário, nem uniforme ou algo que possa ser quantificado. É uma noção de orientação. A sensação de "ser direta" é uma impressão ou um sentimento passado pela interface.

Como base assume-se que a essência da sensação de "ser direta" resulta de um mínimo de fontes cognitivas. Ou vendo por outro lado, a necessidade de fontes de conhecimentos adicionais na utilização de uma interface resulta em uma sensação mais "indireta". A formação do sentimento de "ser direto" é devida a uma adaptação do usuário de modo que o designer da interface nunca poderá

controlar completamente o processo de como fornecer este sentimento de forma precisa [8].

A expressão “Manipulação Direta” foi definida por Ben Shneiderman como uma interação homem-computador que envolve a representação contínua dos objetos de interesse, ações e respostas [9]. A intenção é permitir que o usuário manipule diretamente os objetos apresentados a eles, usando as ações que correspondam similarmente ao mundo físico.

Ter metáforas com o mundo real para objetos e ações facilita para que o usuário aprenda a utilizar uma interface (muitas vezes classificada como uma interface mais natural ou intuitiva), e a resposta contínua permite que o usuário conclua suas tarefas com menos erros e em menos tempo. Isso devido a fato de poder acompanhar os seus resultados antes que a ação tenha sido realmente finalizada. A literatura de psicologia cita as forças das representações visuais nos termos de velocidade e de retenção de aprendizado. A manipulação direta aproveita estas forças tendo por resultado os sistemas cujas operações são fáceis de aprender e se usar e difícil de esquecer

Um exemplo para ilustrar os princípios da manipulação direta em contraste com o estilo de interação intermediada (comandos digitados no teclado tradicional), é um passeio de carro. Com a manipulação direta, o motorista dirige o carro controlando a direção, a roda e os pedais. O carro responde imediatamente a suas ações, e estas respostas são imediatamente evidentes. Se o motorista estiver fazendo um erro tal como girar demasiadamente a direção, ele pode rapidamente reconhecer este erro e executar uma medida corretiva. Com o estilo de interação intermediada, este motorista passa a ser um passageiro e se senta no banco traseiro e passa a dar ordens para outra pessoa que agora dirige o carro. E, este novo piloto possui um fraco relacionamento pessoal e um vocabulário limitado. Agora o passageiro perde a sensação da estrada e a visão da direção para onde está indo. Ele passa a confiar neste motorista que se não receber direções explícitas em uma ordem determinada pode parar no meio do caminho ou terminar em um destino desconhecido [10].

A manipulação direta é muito associada com sistemas WIMP (window, icon, menu, pointing device) pois estes quase sempre incorporam algum tipo de manipulação direta. Entretanto, a manipulação direta não deve ser confundida com estes outros termos, porque não implica o uso das janelas ou mesmo da saída

gráfica. Por exemplo, os conceitos da manipulação direta podem ser aplicados às interfaces para usuários com deficiência visual usando uma combinação de dispositivos e softwares sonoros.

É também possível projetar uma relação de WIMP que intencionalmente não empregue a manipulação direta. Por exemplo, na maioria de versões de interfaces com janelas (por exemplo, Microsoft Windows) os usuários podem reposicionar uma janela arrastando a com o mouse, mas não redesenham continuamente em posições intermediárias durante o arrasto. Ao invés disso, por exemplo, um esboço retangular da janela pode ser extraído durante o arrasto, com os índices completos da janela que são redesenhados somente uma vez que o usuário solta o botão do mouse. Isto era necessário em computadores mais antigos que não possuíam memória e/ou processador suficiente para redesenhar rapidamente dados de uma janela que fosse arrastada [11].

Os sistemas de manipulação direta têm prós e contras. Por exemplo, a resposta imediata, a tradução natural de intenções de ações e a limitação do número de ações físicas que se pode realizar com os dispositivos de entrada, tornam algumas tarefas mais fáceis. A combinação dos vários níveis da linguagem de interface (na semântica direta) aumenta a facilidade e o poder de executar algumas tarefas ao pequeno custo de generalização e flexibilidade. Mas no caso de uma operação repetitiva, esta seria feita com muito mais eficiência através de um script, isto é, com uma descrição simbólica das tarefas que devem ser realizadas [8]. Um outro fator quer pode ser problemático é a limitação tecnológica na interação homem-computador (dispositivos de entrada e saída de dados). Manipulação direta exige dispositivos adaptados para seus usuários, de forma que o sistema consiga captar e obedecer imediatamente às ordens e retornar continuamente as respostas em um formato compreensível pelo usuário.

A princípio a interface fornecida por este framework aqui descrito se preocupa mais com o sentimento de “ser direto”, de manter sempre a continuidade da representação dos objetos e das respostas de ações. Algumas poucas características indiretas, que se tornaram padrões para usuários de computador, foram mantidas como, por exemplo, copiar e colar.

2.4
Programação por Usuário Final

A maior parte das pessoas utiliza computadores como usuários finais de programas empacotados. Infelizmente os escritores destes programas não podem saber os detalhes das tarefas que cada usuário está tentando fazer. Tentando atender as necessidades de usuários diversos, os desenvolvedores montam seus programas com centenas das características que a maioria dos usuários nunca utiliza. Isto seria mais simples se cada usuário final pudesse adicionar as funções que bem entendesse.

Fornecer esta potencialidade em um programa não é trivial. Os programas devem ser projetados a aceitar componentes escritos pelos usuários em lugares apropriados. Deve haver uma maneira armazená-los e controlá-los. Visto que a maioria dos usuários não tem tempo ou a inclinação de aprender as ferramentas e as habilidades de um programador profissional, alguns requisitos são necessários. A expressividade e generalidade da programação são trocadas para usabilidade utilizando uma variedade de metáforas. A programação pode ser feita muito mais facilmente através de metáforas – símbolos matemáticos; seqüência de ações de uma GUI; diagrama de circuito; linguagem específica de aplicação – do que com programação convencional.

Porque as metáforas, com suas potencialidades e limitações, diferem extensamente dependendo do usuário e suas funcionalidades. Não há um método de programação por usuário final. Ao invés disso há uma variedade de técnicas, tais como a programação por demonstração, programação visual, e muitas linguagens específicas de domínio. Idealmente há uma progressão suave de metáforas simples, mas limitadas, até técnicas mais complexas e mais poderosas como os avanços do usuário-programador [12].

Assim, a programação por usuário final tem como objetivo a busca de uma forma de possibilitar aos usuários finais obter melhor aplicabilidade e usabilidade de um software. Aplicabilidade aqui, diz respeito às condições sob as quais um artefato (uma função, uma ferramenta, etc.) poderia ser usado e usabilidade é vista de acordo com os critérios definidos por conforme a citação abaixo:

O critério chave de usabilidade de um sistema é o quanto de extensão ele suporta para que usuários que trabalham com ele possam: entender, aprender e modificar [13].

Para um software atingir um alto grau de usabilidade é necessário que o designer do software o projete de modo que ele sustente o aprendizado por parte dos usuários em dois níveis:

- Aprendizado através do qual o usuário vem a entender como o software funciona;
- Aprendizado através do qual o usuário descubra como adaptar e estender a tecnologia de forma a satisfazer suas demandas e as contingências de seu trabalho.

A idéia central dos pesquisadores na área de programação por usuário final é permitir aos usuários finais modificar suas aplicações para que estas sirvam melhor às suas necessidades e/ou estilos de trabalho. Estas idéias surgiram na academia e foram de certa forma, incorporadas pela indústria de software, com maior ou menor sucesso, em produtos bem conhecidos [14].

Possibilitar a um usuário final personalizar suas tarefas certamente aumenta o potencial de aplicabilidade e de usabilidade de uma aplicação. Contudo, isto requer que o usuário possa realizar algum tipo de alteração na sua programação. Estas alterações de funcionalidade da aplicação são denominadas extensões e podem ser realizadas basicamente de duas formas [14]:

- Através do uso da linguagem de interface da aplicação ou
- Através do uso de uma linguagem de extensão que opere embutida na aplicação.

Neste trabalho existem dois tipos de usuários: o usuário desenvolvedor e o usuário final. O usuário final é todo usuário que apresente alfabetização computacional suficiente para operar uma aplicação de software para a realização de suas tarefas de modo eficaz. Assim, queremos deixar claro que, nesta abordagem, não consideramos necessário que o usuário tenha a priori qualquer conhecimento de.

Neste framework o usuário final tem a possibilidade, de através de um arquivo de configuração, adicionar filtros que restringem a visão sobre as

interfaces e alterar a semântica de suas ações. Estes a partir de opções implementadas pelo usuário desenvolvedor.

2.5 Frameworks

Em programação, framework é um termo usado geralmente para se referir a um conjunto de classes e bibliotecas que são utilizadas para implementar a estrutura de uma aplicação. Empacotando uma grande quantidade de código reusável em um framework, muito tempo dos desenvolvedores é economizado já que estes ganham o tempo que seria gasto na tarefa de reescrever o código padrão para cada nova aplicação desenvolvida. [15]

Os frameworks tornaram-se populares com a ascensão de interfaces gráficas de usuário (GUI), já que estas tendem a promover uma estrutura padrão para aplicações. É também muito mais simples criar ferramentas automáticas de criação de GUI quando um framework padrão é utilizado, já que o código da camada inferior da aplicação é previamente conhecido.

Um framework captura as funcionalidades comuns a várias aplicações pertencentes a um mesmo domínio. Ele provê uma solução para uma família de problemas semelhantes usando um conjunto de classes e interfaces. Estas classes possuem objetos que colaboram para cumprir suas responsabilidades. Desta forma o conjunto de classes deve ser flexível e extensível para permitir a construção de várias aplicações com pouco esforço, especificando apenas as particularidades de cada aplicação. Um framework é uma aplicação quase completa, mas com pedaços faltando. A partir de um framework o trabalho consiste apenas em prover os pedaços que são específicos para uma aplicação.

A principal diferença entre um framework e uma biblioteca de classes OO é que em uma biblioteca de classes, cada classe é única e independente das outras e em um framework, as dependências/colaborações estão embutidas.

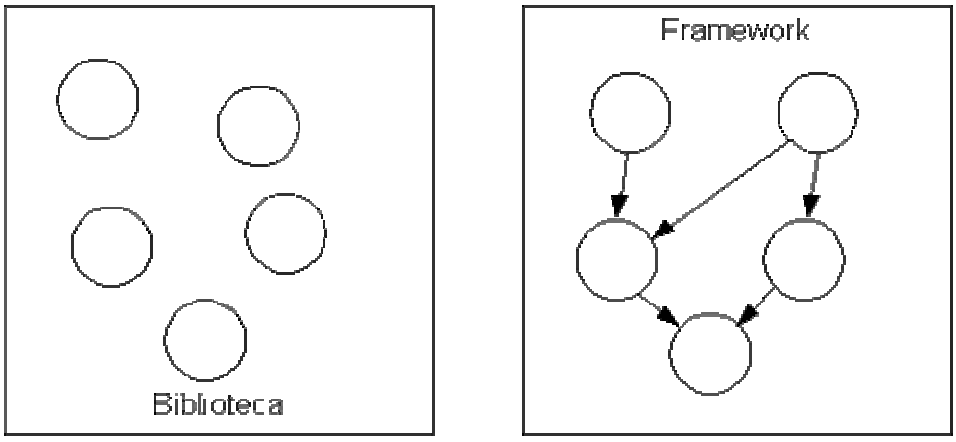

Figura 2 – Biblioteca e Framework

Vê-se, portanto, que um framework impõe um modelo de colaboração (o resultado da análise e design) ao qual o desenvolvedor deve se adaptar. Visto que a comunicação entre objetos já está definida, o projetista de aplicações não precisa saber quando chamar cada método: é o framework que faz isso. Em uma biblioteca de classes não se pode embutir conhecimento do domínio (análise + design) e com biblioteca, as aplicações ficam responsáveis por criar as colaborações [17].

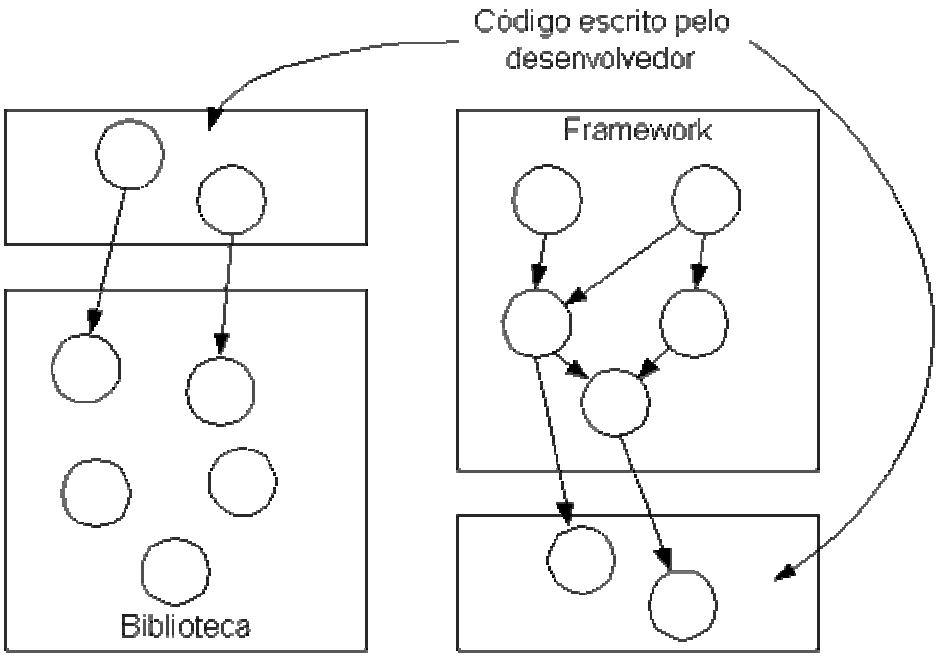

Figura 3 – Extensão de Bibliotecas e Frameworks

Como características básicas, o framework deve ser reutilizável de forma a ser aproveitado em mais de um contexto e extensível, podendo agregar informações e ações adicionais. Obviamente, tudo isto deve ser feito sem perder a eficiência.

Das vantagens da elaboração e utilização de um framework a principal é a redução de tempo e custo para o desenvolvimento de aplicações. A possibilidade de reutilização permite aos desenvolvedores sempre agregar novos valores. Finalmente, a utilização de um framework fornece melhor consistência e compatibilidade entre aplicações.

2.6 Modelo de Implementação

Da mesma forma que foi definida na dissertação do Leonardo Belmonte parte desta secção é comum em ambas as teses. Para este framework, o modelo de implementação ideal é o MVC (*Model View Controller*). O MVC é uma arquitetura de software que separa o modelo de dados da aplicação, a interface do usuário, e o controle de lógica em três componentes distintos. Dessa forma, alterações feitas em um componente causa impactos mínimos nos outros.

MVC muitas vezes é confundido com um design pattern para software. Entretanto, o MVC se assemelha mais a uma arquitetura de aplicação do que a um típico design pattern. Então, o termo *architectural pattern* pode ser utilizado.

Para a construção de uma aplicação utilizando uma arquitetura MVC devem ser considerados três módulos:

Model (Modelo)

Representação da informação para um domínio específico no qual a aplicação é executada. Modelo é apenas um outro nome para a camada de domínio. Domínio lógico acrescenta significado aos dados.

View (Visualizador)

Renderiza o modelo em uma forma adequada para a interação, tipicamente um elemento da interface do usuário.

Controller (Controlador)

Representa eventos, geralmente ações do usuário, e realiza mudanças no componente *Model* e às vezes no *View*.

Muitas aplicações utilizam repositórios de dados como, por exemplo, bancos de dados. MVC não menciona essa camada de dados, pois considera que ela está encapsulada pelo componente *Model*.

Um fluxo de controle geralmente funciona da seguinte forma:

1. O usuário interage com a interface de alguma forma, pressionando um botão, por exemplo.
2. Um controlador trata o evento vindo da interface.
3. O componente *controller* acessa o componente *model*, possivelmente atualizando-o de uma maneira compatível com a ação do usuário.
4. O componente *view* utiliza o componente *model* para gerar uma interface para o usuário apropriada. O *view* pega os dados do *model*. O *model* não tem conhecimento direto do *view*.
5. A interface do usuário espera por novas interações do usuário, a qual começa o ciclo novamente.

O MVC introduz o objeto *controller* entre o *view* (GUI) e o *model* (objeto) para fazer a comunicação desses dois objetos. A implementação atual do objeto *controller* pode variar, mas a idéia de que o objeto transforma eventos em mudanças nos dados e na execução dos métodos deve ser considerada como a essência desse *pattern*.

O diagrama abaixo mostra a relação existente entre os componentes. A linha sólida indica associação direta e a pontilhada indica associação indireta.

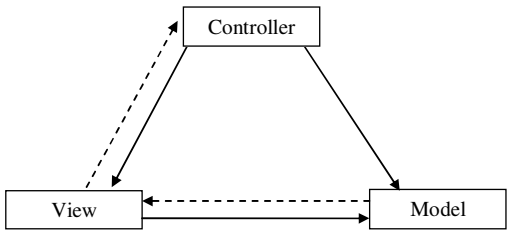

Figura 4 – Modelo MVC

A interface gerada por este framework se encaixa apenas como uma *view* de um modelo e um *controller*. Acoplada a um modelo e a um controlador, estes três componentes formam uma única solução baseada na arquitetura MVC. Exemplos de funcionamento do framework serão demonstrados no capítulo 5 onde será utilizado o modelo desenvolvido por Leonardo Belmonte.