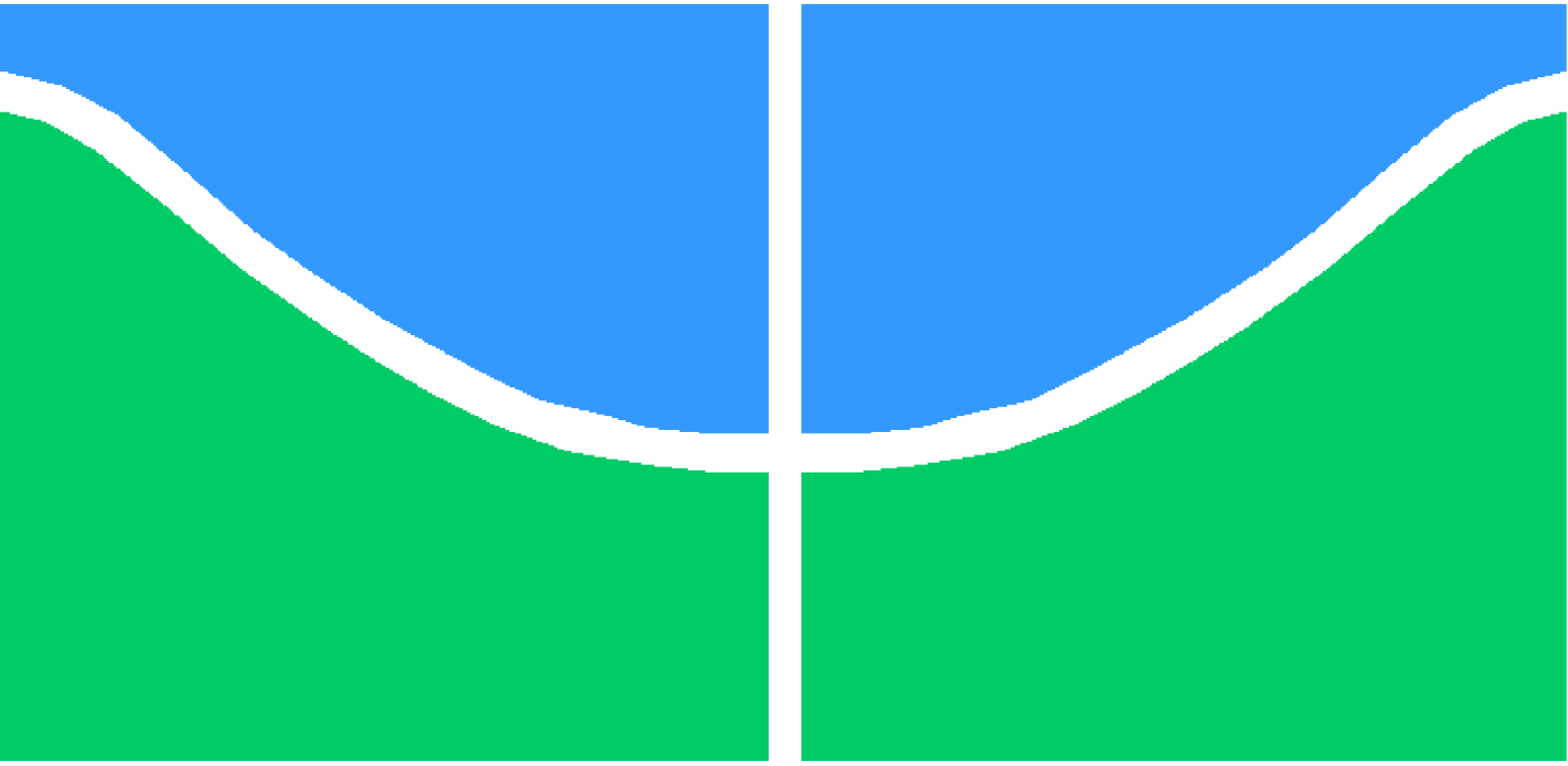

Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Engenharia de Software

ACE-CAST: Uma Ferramenta de Apoio à Argumentação Colaborativa.

Autor: André Cruz Alves Cavalcante
Orientador: Prof. Dr. Maurício Serrano
Coorientador: Prof. Dr.ª Milene Serrano

Brasília, DF
2014

André Cruz Alves Cavalcante

ACE-CAST: Uma Ferramenta de Apoio à Argumentação Colaborativa.

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Prof. Dr. Maurício Serrano

Coorientador: Prof. Dr.ª Milene Serrano

Brasília, DF

2014

André Cruz Alves Cavalcante
ACE-CAST: Uma Ferramenta de Apoio à Argumentação Colaborativa./ André Cruz Alves Cavalcante. – Brasília, DF, 2014-
99 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Maurício Serrano

Coorientador: Prof. Dr.ª Milene Serrano

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2014.

1. Argumentação. 2. Ferramenta Colaborativa. I. Prof. Dr. Maurício Serrano. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. ACE-CAST: Uma Ferramenta de Apoio à Argumentação Colaborativa.

CDU 02:141:005.6

André Cruz Alves Cavalcante

ACE-CAST: Uma Ferramenta de Apoio à Argumentação Colaborativa.

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, :

Prof. Dr. Maurício Serrano
Orientador

Prof. Dr.ª Milene Serrano
Coorientador

Prof. Dr. Edson Alves
Convidado 1

Prof. Dr. Wander Cleber Pereira
Convidado 2

Brasília, DF
2014

Agradecimentos

Aos meus pais, pelo amor, incentivo e apoio incondicional.

A Universidade de Brasília, pelo ambiente criativo e excelência acadêmica.

Aos meus orientadores, pelo suporte motivacional e técnico na elaboração deste trabalho.

Aos amigos que fizeram parte da minha formação acadêmica e que vão continuar presentes em minha vida.

A mente que se abre a uma nova ideia
jamais voltará ao seu tamanho original.
(Oliver Wendell Holmes)

Resumo

A argumentação possui um papel essencial na sociedade. Diariamente, as pessoas participam de atividades em que o ato de argumentar é constante. Seja ela para defender uma ideia ou para tomar uma decisão. Apesar da importância, várias ferramentas permitem apenas a representação textual dos elementos de uma argumentação. A avaliação da aceitabilidade dos argumentos fica a critério da interpretação de cada participante. O objetivo deste trabalho é o desenvolvimento de uma ferramenta web colaborativa que facilite o processo de argumentação online. Esta ferramenta deve permitir que as pessoas possam estruturar e verificar a validade de seus argumentos de forma gráfica em tempo real. Para isto, um processo de desenvolvimento de software baseado em princípios ágeis foi elaborado. Ao término deste processo, a ferramenta ACE-CAST[1] foi implementada. Esta ferramenta permite a representação de uma argumentação através de grafos que são atualizados e avaliados em tempo real na medida em que os participantes adicionam novas informações. Para obter as impressões iniciais dos usuários que testaram a ferramenta ACE-CAST, um cenário de uso baseado na técnica de pesquisa-ação foi definido e executado.

Palavras-Chaves: argumentação. discussão. ferramenta colaborativa.

[1] O nome é baseado no framework ACE: (base de conhecimento para o desenvolvimento da ferramenta) e a sigla CAST (Collaborative Argumentation Support Tool) ou (Ferramenta de Suporte a Argumentação Colaborativa.)

Abstract

The argument has an essential role in society. Every day, people participate in activities in which the act of arguing is constant. Whether to defend an idea or to make a decision. Despite its importance, several tools allow only the textual representation of the elements of an argument. The evaluation of the acceptability of arguments is up to the interpretation of each participant. The objective of this work is the development of a collaborative web tool that facilitates the online process of argumentation. This tool should allow multiple people to structure and evaluate their arguments in real time. For this, a software development process based on agile principles was prepared. At the end of this process, the tool ACE-CAST[2] was developed. This tool allows the representation of the argument in graphs that are updated and evaluated in real time as participants add new information. Finally, a use scenario based on action research technique was performed to obtain the initial impressions of users who tested the tool ACE-CAST.

Key-words: argumentation. discussion. collaborative tool.

[2] The name is based on the ACE framework: (knowledge base for the development of the tool) and the acronym CAST (Collaborative Argumentation Support Tool)

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

MVC	Model-View-Controller
Grails	Groovy on Rails
TCC	Trabalho de Conclusão de Curso
UnB	Universidade de Brasília
RDF	Resource Description Framework
OWL	Web Ontology Language
ACE	Acceptability Evaluation Framework
IBIS	Issue-Based Information System
AIF	Argument Interchange Format

Sumário

1 Introdução

A argumentação possui um papel essencial na sociedade. Diariamente diversas pessoas participam de discussões, negociações, deliberações e várias outras atividades colaborativas em que o ato de argumentar é constante. A argumentação possibilita às pessoas ferramentas para que elas possam defender seus interesses e crenças. De acordo com (SCHNEIDER; GROZA; PASSANT, 2013), a argumentação simboliza o estudo das opiniões e pontos de vistas expressos por humanos com o intuito de gerar conclusões através de raciocínio lógico.

Segundo (EEMEREN; GROOTENDORST; HENKEMANS, 1996), a argumentação pode ser vista como uma atividade verbal, social e causal. É verbal devido a necessidade de representação dos argumentos em uma linguagem ordinária, seja ela oral ou escrita. Meios não verbais como gestos e expressões faciais são válidos, porém não substituem as expressões verbais; É social devido a participação de diversos interlocutores expressando suas ideias, considerando os pontos de vistas alheios e tomando decisões com base nessas informações; É racional, pois o argumento apresentado por um interlocutor possui uma percepção coerente com base na posição defendida.

Devido a aplicação em diversos contextos, a argumentação não é restrita a apenas uma disciplina. A prática da argumentação está presente e é objeto de estudo em várias áreas, tais como a inteligência artificial, engenharia de requisitos, tomada de decisões, psicologia, lingüística, filosofia, teoria legal e lógica (CHARWAT et al., 2013).

Nos últimos 50 anos a argumentação vem sendo aprimorada. Diversos modelos de argumentação foram propostos. O modelo de Toulmin (TOULMIN, 1958), proveniente da filosofia, representa um marco na história da argumentação. Este modelo permitiu a ruptura do paradigma formal de argumentação, sugerindo uma maneira flexível de argumentar a partir de conceitos informais. Este modelo é utilizado em diversas situações, tais como conversas informais, deliberações legais e debates científicos. O modelo de Toulmin tem como meta a representação e análise dos elementos potenciais que constituem qualquer tipo de argumentação, de modo que seja possível gerar uma estrutura menos ambígua (TOULMIN, 1958). Os elementos que fazem parte deste modelo são os dados, conclusões, justificativas, garantias, modalidades, suportes e refutações.

Após o modelo Toulmin, outra contribuição importante para a argumentação ocorreu em 1970. Neste ano foi publicado o IBIS[1]. O IBIS é um esquema de argumentação criado para coordenar e planejar o processo de tomada de decisões (KUNZ et al., 1970). Ele foi originalmente criado como um sistema de documentação destinado a organiza-

[1] Issue-Based Information System ou Sistemas de Informação Baseados em Questões

ção da informação e a subseqüente consulta para facilitar a compreensão das decisões que foram tomadas (SCHNEIDER; GROZA; PASSANT, 2013). No esquema IBIS a argumentação é estruturada por meio de questões, em que o problema é apresentado, posições, onde são propostas as alternativas para a resolução dos problemas, e argumentos, que qualificam uma ou mais posições.

Os avanços computacionais permitiram que diversas ferramentas fossem desenvolvidas com base nos modelos de argumentação citados acima. No mercado existem diversas soluções que seguem alguns princípios do modelo de Toulmin: Araucaria (REED; ROWE, 2004), Rationale, Debatabase, entre outros. O IBIS também possui algumas opções no mercado tais como a Compendium e a QuestMap.

As soluções baseadas no modelo de Toulmin apenas permitem a diagramação e visualização dos argumentos. A lógica racional para determinar a aceitabilidade de um argumento partia da subjetividade dos participantes da discussão. Quando os elementos da argumentação aumentam, a percepção de aceitabilidade de determinado argumento fica complexa (TOULMIN, 1958). Já as soluções baseadas no IBIS oferecem suporte ao processo de tomada de decisões. Porém, assim como o modelo de Toulmin, o IBIS não define um critério de aceitabilidade para verificar a validade de seus elementos. Esta percepção fica a critério dos participantes da discussão.

Na ciência da computação, existe a preocupação para identificar critérios formais para determinar a aceitabilidade dos argumentos (RAHWAN, 2008). Uma das principais propostas, proveniente da inteligência artificial, foi o framework de argumentação abstrata proposto por Dung (DUNG, 1995). O framework é composto por entidades abstratas (argumentos) e relações binárias de ataque ou derrota entre eles. O uso destes elementos irá gerar um grafo caracteriza os conflitos entre os argumentos. A aceitabilidade deste framework é obtida através de conjuntos compostos por argumentos que não se atacam e/ou se defendem. Esse tipo de aceitabilidade é conhecida como *aceitabilidade conjunta*.

Jureta, Mylopoulos e Faulkner (2009) desenvolveram o ACE[2] que também oferece critérios para a identificação e avaliação da aceitabilidade dos argumentos no contexto da engenharia de requisitos. Este framework é composto por uma linguagem para representar as informações relevantes de uma discussão e algoritmos para recuperar argumentos relevantes e avaliar sua aceitabilidade. A discussão é representada através de um grafo. A informação é representada como proposições, outros elementos como conflitos, regras de inferência e preferências aumentam a semântica do modelo. O esquema de aceitabilidade do ACE é baseado em rotular cada elemento da argumentação individualmente. Esse tipo de aceitabilidade é conhecido como *aceitabilidade individual*, cada argumento é rotulado com base nos elementos conectados com o mesmo.

[2] Acceptability Evaluation framework ou framework de avaliação da aceitabilidade

Diversas ferramentas foram implementadas utilizando o framework de Dung. Uma das principais é a ArguLab (PODLASZEWSKI; CAMINADA; PIGOZZI, 2011), que permite a construção de um grafo de argumentos e oferece a avaliação do grafo resultante utilizando as relações semânticas presentes no modelo de Dung. Apesar de ter automatizado a aceitabilidade dos argumentos a ferramenta é complexa e apenas usuários com conhecimentos avançados em argumentação são capazes de utilizá-la. Ferramentas baseadas no ACE não foram encontradas.

1.1 Problemas

A internet encoraja a colaboração e a socialização e tem facilitado a propagação da informação, superando barreiras geográficas. A internet permite várias formas de comunicação, mas uma característica importante que todas essas formas tem em comum é a necessidade de um sistema que ofereça apoio, análise e suporte eficiente para auxiliar as atividades colaborativas (DANIIL; DASCALU; TRAUSAN-MATU, 2012).

Várias tecnologias oferecem suporte a argumentação, como listas de e-mail, sistemas de auxílio à tomada de decisões, sistemas de negociações, entre outras. Entretanto, várias destas tecnologias citadas não funcionam bem na prática (MOOR; AAKHUS, 2006). O problema é que o conhecimento gerado por essas tecnologias dificilmente é utilizado para inferir conclusões lógicas através de processamento computacional. Com base no estudo realizado, observou-se que o motivo deste problema é a falta de adoção de modelos formais. Na maioria das vezes utiliza-se apenas texto para representar a argumentação. A semântica necessária para a realização de inferências é difícil de ser estruturada.

Para permitir a análise computacional, diversos frameworks de argumentação são representados através de grafos (DUNG, 1995) (JURETA; MYLOPOULOS; FAULKNER, 2009). Apesar de ser a escolha racional, diversos usuários sem conhecimento de lógica possuem dificuldade de entender e raciocinar acerca dos modelos gerados. Outro aspecto que deve ser considerado é a dificuldade no desenvolvimento de sistemas baseados em grafos. Os algoritmos associados bem como a interface gráfica para abstrair a complexidade para o usuário requer cuidados especiais.

Através das pesquisas realizadas é possível visualizar lacunas nas ferramentas de argumentação investigadas. Várias delas não oferecem suporte amplo ao processo de argumentação (diagramação, visualização e avaliação) e quando oferecem, a usabilidade da ferramenta deixa a desejar (GONZÁLEZ et al., 2010).

1.2 Objetivos

Para obter êxito neste trabalho de conclusão de curso, os seguintes objetivos gerais e específicos foram definidos.

1.2.1 Objetivos Gerais

1. Desenvolver uma ferramenta web colaborativa que auxilie o processo de argumentação. Esta ferramenta deve permitir que diversos usuários possam estruturar suas discussões e avaliar a aceitabilidade de cada argumento em tempo real;

1.2.2 Objetivos Específicos

1. Fazer uma avaliação dos modelos de argumentação relevantes. A avaliação será utilizada para obter as melhores práticas de cada modelo. Desta forma, possibilitará uma elicitação de requisitos consistente para a ferramenta que será construída.

2. Avaliar as ferramentas de discussão disponíveis na internet. O resultado desta avaliação permitirá a identificação de boas práticas de projeto e implementação, bem como a identificação de problemas que poderão ser evitados no desenvolvimento da nova ferramenta;

3. Realizar a engenharia de requisitos da ferramenta a ser desenvolvida utilizando técnicas orientadas à meta;

4. Gerenciar e executar o processo de desenvolvimento da ferramenta seguindo princípios da metodologia ágil.

1.3 Justificativa

Utilizando as ferramentas de discussão online disponíveis, nem sempre todos os usuários possuem a percepção das informações presentes em uma discussão (MOOR; AAKHUS, 2006). Não conhecendo bem as informações, os usuários acabam tomando decisões erradas que prejudicam a natureza lógica da discussão. Outro aspecto importante é a rastreabilidade das premissas que geraram uma opinião. Caso um usuário queira saber a motivação de uma opinião, o mesmo terá que analisar todas as informações de uma discussão buscando relações estabelecidas previamente.

Em um contexto empresarial, onde que é necessário ter pleno controle das decisões tomadas, qualquer tipo de imprecisão acerca de uma informação pode ser fatal. Podendo resultar em perdas graves aos ativos de uma organização.

Ao término deste trabalho, as pessoas terão acesso a uma ferramenta que irá conectar sintaticamente os argumentos e facilitará a análise lógica das informações presentes em uma discussão. Com esta ferramenta, as seguintes premissas são oferecidas aos usuários:

- Verificação automática de suas opiniões. Deste modo, o usuário saberá a validade de seus argumentos sem a necessidade de ler todas as informações de uma discussão;
- Conexão com outros usuários através de vínculos de amizade ou grupos;
- Interface Gráfica com elementos assíncronos. Todas as informações geradas por outros usuários que estão participando de uma discussão serão visualizadas em tempo real.

1.4 Metodologia

Para guiar o desenvolvimento deste Trabalho de Conclusão de Curso, foi definida uma metodologia composta de quatro etapas que serão descritas a seguir:

1. A primeira etapa consistiu na realização do Referencial Teórico com intuito de obter informações acerca da teoria da argumentação e de tecnologias de desenvolvimento.
2. A segunda etapa consistiu na especificação da solução que será desenvolvida. Nesta etapa foram definidos os modelos estruturais e comportamentais que nortearam o desenvolvimento da solução.
3. A terceira etapa consistiu no desenvolvimento da solução. Nesta etapa foi definido o *Backlog* do produto e as *Sprints* de desenvolvimento. A partir das *Sprints* diversos incrementos do produto foram desenvolvidos até a obtenção do produto final.
4. A quarta etapa consistiu em um cenário de uso baseado na técnica de pesquisa-ação. A ferramenta foi utilizada por grupo de usuários e suas impressões iniciais foram capturadas com base em uma ficha de avaliação.

1.5 Organização do Trabalho

No Capítulo 2 - *Teoria da Argumentação* são apresentados os conceitos referentes a argumentação em diferentes contextos. Para isto, é feita uma análise conceitual da argumentação. A seguir, apresenta-se os modelos e frameworks relevantes para a prática da argumentação. Para finalizar, apresenta-se uma ontologia que descreve os elementos fundamentais de uma argumentação.

No Capítulo 3 - *Metodologias e Tecnologias* são apresentados as metodologias e tecnologias utilizados para o desenvolvimento deste trabalho.

No Capitulo 4 - *Processo Metodológico* são apresentados os passos necessários para a conclusão do trabalho. Para isto, apresenta-se as descrições das tarefas que serão executadas bem como a metodologia de desenvolvimento utilizada neste trabalho.

No Capítulo 5 - *Resultados* são apresentados os resultados obtidos com este trabalho. Entre os resultados, destacam-se a elicitação de requisitos utilizando técnicas orientadas à meta, a arquitetura de software, a ferramenta desenvolvida ACE-CAST e um cenário de uso para testar a solução.

No Capítulo 6 - *Considerações Finais* são apresentadas as conclusões, próximos passos, e contribuições obtidas ao término deste trabalho.

2 Teoria da Argumentação

Neste capítulo são apresentados os conceitos relativos a argumentação. Deste modo, define-se o conceito de argumentação, bem como seu processo. Logo após, é realizada uma análise envolvendo vários modelos e frameworks de argumentação. Para finalizar, apresenta-se uma ontologia que descreve os elementos fundamentais de uma argumentação.

2.1 Argumentação

A argumentação é uma forma vital de cognição humana. Diariamente, pessoas são confrontadas com informações conflitantes e forçadas a lidar com situações inconsistentes (BESNARD; HUNTER, 2008). O uso da argumentação representa uma atividade de comunicação essencial para a sociedade. Existem diversas tecnologias que oferecem suporte a essa prática, tais como listas de e-mails, sistemas de suporte a tomada de decisões e sistemas de suporte a negociação (MOOR; AAKHUS, 2006). Apesar da diversidade, as boas práticas de uma argumentação nem sempre estão presentes nessas tecnologias. Geralmente estes mecanismos desencorajam o debate e facilitam a argumentação de baixa qualidade e o pensamento devoluto (BEX et al., 2013).

Segundo Rahwan (2005), a argumentação pode ser vista como uma interação social baseada em princípios. Ela é composta de argumentos incompatíveis e almeja chegar a uma conclusão consistente e racional. Uma das principais metas da argumentação é a resolução de pontos de vistas controversos. Estes pontos de vistas devem ser justificáveis ou refutáveis dependendo da informação disponibilizada. Isto distingue a argumentação do raciocínio dedutivo clássico, em que o acréscimo de novas premissas não influencia uma prova previamente definida.

Já Besnard e Hunter (2008) definem a argumentação como sendo um processo em que argumentos e contra-argumentos são construídos e manipulados. A manipulação dos argumentos envolve a comparação, avaliação e julgamento da aceitabilidade dos mesmos com base em critérios definidos.

A argumentação é uma atividade verbal, social e racional. É verbal devido à necessidade de representação dos argumentos em uma linguagem ordinária, seja ela oral ou escrita. Meios não verbais como gestos e expressões faciais podem ser considerados, porém não substituem as expressões verbais; É social devido à participação de diversos interlocutores expressando suas ideias, considerando os pontos de vistas alheios e tomando decisões com base nessas informações; É racional, pois o argumento apresentado por um

interlocutor possui uma percepção racional a partir da posição que o mesmo defende (EEMEREN; GROOTENDORST; HENKEMANS, 1996).

A teoria da argumentação possui dimensões descritivas e normativas. Ela é descritiva, pois descreve práticas argumentativas de um discurso empiricamente. Os principais profissionais envolvidos no estudo desta dimensão possuem experiência em áreas como psicologia e análise lingüística do discurso; Ela é normativa, pois deve refletir criticamente a racionalidade de um discurso. Os principais envolvidos com essa dimensão são profissionais dos ramos da lógica e filosofia. A divergência entre as abordagens normativas e descritivas podem gerar certos equívocos na teoria da argumentação. Entretanto, uma teoria da argumentação concisa requer o uso destas duas dimensões (EEMEREN; GROOTENDORST; HENKEMANS, 1996).

O processo de argumentação pode ser definido com base em quatro tarefas fundamentais: identificação, análise, avaliação e invenção (WALTON, 2009). Apresenta-se na Tabela 1 a descrição de cada uma destas tarefas.

Tabela 1 – Etapas da Argumentação.

Tarefa	**Descrição**
Identificação	Essa tarefa tem como objetivo identificar premissas e conclusões de uma comunicação verbal seja ela escrita ou verbal. A identificação dos argumentos pode ser feita de várias maneiras. Dependendo do esquema de argumentação escolhido, a relação entre os argumentos pode mudar.
Análise	Essa tarefa tem como objetivo encontrar premissas ou conclusões nos argumentos que não estão explicitas. Novas informações auxiliam na avaliação dos argumentos.
Avaliação	Essa tarefa tem como objetivo avaliar a força ou fraqueza dos argumentos com base em critérios definidos.
Invenção	Essa tarefa tem como objetivo a construção de novos argumentos para provar conclusões específicas.

Atualmente, os modelos de argumentação têm sido utilizados em diversas áreas tais como gestão do conhecimento, elaboração de provas e teoremas, inteligência artificial, lógica de programação, sistemas jurídicos, sistemas de tomada de decisões e negociações (BENTAHAR; MOULIN; BéLANGER, 2010). Devido ao caráter multidisciplinar da ar-

gumentação, diversas abordagens foram propostas nos últimos 60 anos. Entre as principais contribuições destacam-se:

- O modelo de Toulmin (TOULMIN, 1958) representa uma alternativa à lógica clássica tradicional, bem mais adequada para lidar com a argumentação habitual utilizada diariamente. Este modelo tem como metas a identificação e a avaliação crítica dos elementos principais de uma argumentação. Para isto foi elaborado um conjunto de elementos para estruturar uma argumentação de forma concisa;

- O IBIS[1] (KUNZ et al., 1970) framework oferece suporte ao planejamento e coordenação do processo de tomada de decisões. O IBIS auxilia na identificação, estruturação e resolução de questões levantadas por uma equipe.

- Dung (1995) desenvolveu um framework de argumentação abstrato onde o foco era definir formalmente a aceitabilidade dos argumentos. O framework consiste em um conjunto de argumentos e relações binárias representando a situação de ataque entre estes argumentos. Neste ponto, um argumento é uma estrutura atômica cujo papel é definido a partir das relações com outros argumentos. Preocupações quanto à estrutura interna do argumento são desconsideradas.

- O ACE[2] (JURETA; MYLOPOULOS; FAULKNER, 2009) é um framework de argumentação utilizado na engenharia de requisitos. O objetivo deste modelo é verificar a validação relativa dos artefatos discutidos em uma reunião envolvendo os stackholders e os engenheiros de requisitos. O ACE consiste de uma linguagem para representar informações obtidas a partir de uma discussão, uma condição de aceitabilidade que denota uma decisão comum dos participantes acerca de um artefato e algoritmos para checar automaticamente a condição de aceitabilidade nas discussões.

O modelo de Toulmin e o IBIS framework possuem características descritivas, pois o foco principal destes modelos é a representação de uma argumentação. Enquanto que os frameworks ACE e Dung, além da representação dos argumentos, oferecem regras semânticas formais que permitem a realização de inferências computacionalmente. Estes modelos possuem características normativas e descritivas. Uma análise sucinta de cada modelo é realizada na sessão 2.2. A análise consiste na definição conceitual e na realização de um estudo de caso para verificar a eficácia de cada estratégia.

[1] Issue Based Information System ou Sistema de Informação Baseados em Questões
[2] Acceptability Evaluation Framework ou Framework de Avaliação da Aceitabilidade

2.2 Modelos de Argumentação

Os modelos de argumentação oferecem um conjunto de abstrações e relacionamentos em que elementos de uma discussão podem ser documentados e associados. Um padrão estruturado e sistemático de comunicação é estabelecido (RELVAS; ANTUNES, 2006). A seguir são apresentados os modelos e frameworks de argumentação analisados neste trabalho.

2.2.1 Modelo de Toulmin

O modelo de Toulmin (TOULMIN, 1958) oferece uma alternativa a lógica dedutiva tradicional. A principal preocupação deste modelo é a observação do uso da argumentação na prática em vez da formalização de padrões que tem como objetivo a justificativa dos argumentos (BENOIT; HAMPLE, 1992). Este modelo não se aplica apenas a atividades colaborativas envolvendo um grupo de pessoas, mas também reflexões individuais para alcançar uma conclusão a partir das informações disponível (HITCHCOCK, 2005).

A argumentação prática obteve diversos avanços com a proposta de Toulmin: os argumentos passaram a possuir estruturas bem definidas; os componentes de um argumento bem como seus relacionamentos foram estabelecidos; a adoção dos componentes de um argumento em um ou vários contextos diferentes; e demonstrou que é possível utilizar a mesma estrutura de informação em argumentos diferentes (GASPER; GEORGE, 1998).

De acordo com Toulmin (1958), seis elementos básicos podem ser encontrados em qualquer tipo de argumentação. Apresenta-se na Tabela 2 uma descrição sucinta de cada elemento.

Tabela 2 – Elementos do Modelo de Toulmin.

Elemento	**Descrição**
Conclusão ou Alegação	Posições ou reivindicações sendo alvo de discussões. Representa o objeto alvo da argumentação.
Dados ou Fatos	Motivos ou evidências que ofereçam suporte a consistência da conclusão.
Garantia	Caso os fatos não justifiquem a conclusão, a garantia pode ser utilizada como uma suposição implícita que funciona como uma ponte entre a conclusão e os dados.
Apoio ou Suporte	Oferece auxílio para provar que a garantia definida é verdadeira. Geralmente normas, leis e padrões são utilizados para elaborar este elemento.

Continua na Próxima Página

Tabela 2 – *Continuação da Página Anterior*

Elemento	**Descrição**
Qualificação	Mesmo com os dados e garantias verdadeiros, nem sempre a conclusão é válida. Os qualificadores são utilizados para limitar a força do argumento e/ou para definir condições onde o argumento é verdadeiro.
Refutação	Representa um contra-argumento oriundo de pontos de vista diferentes. Demonstra restrições onde que a conclusão não pode ser aplicada.

Os três primeiros elementos: conclusão, dados e garantia representam os elementos essenciais da argumentação prática. O apoio, qualificação e a refutação são optativos, ou seja, o uso depende do contexto onde o modelo está sendo aplicado (KARBACH, 1987). Apresenta-se na Figura 1 a estrutura básica e o relacionamento entre os elementos de um argumento seguindo as diretrizes propostas por Toulmin.

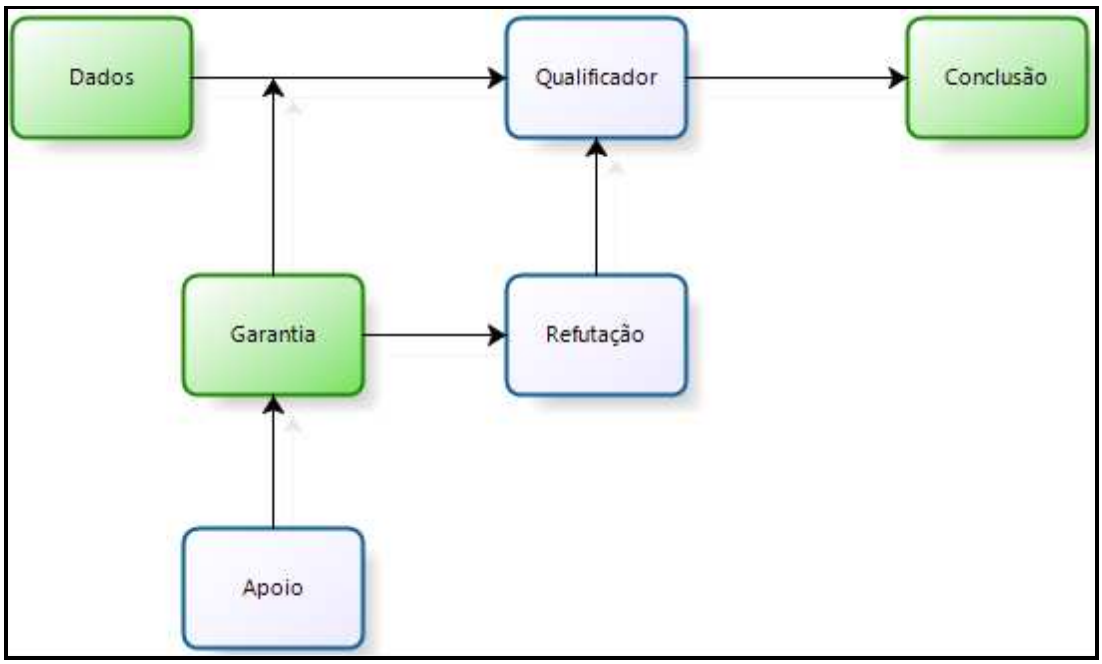

Figura 1 – Estrutura do Modelo de Toulmin.

Com base na Figura 1, os elementos com a cor verde são obrigatórios, enquanto que os elementos de cor azul são opcionais. Os dados representam as bases de raciocínio que suportam a conclusão. As informações implícitas que relacionam os dados e a conclusão são apresentadas através das garantias. Caso existam potenciais objeções à conclusão tomada, qualificadores podem ser definidos. Geralmente os qualificadores são inferidos a

partir de determinadas restrições ou refutações não consideradas pelos dados e garantias do modelo. As refutações também podem ser relacionadas com garantias para aumentar sua autonomia. Os elementos de apoio oferecem justificativas para as garantias às tornando mais confiáveis. Um exemplo da utilização do modelo de Toulmin é apresentado na Figura 2.

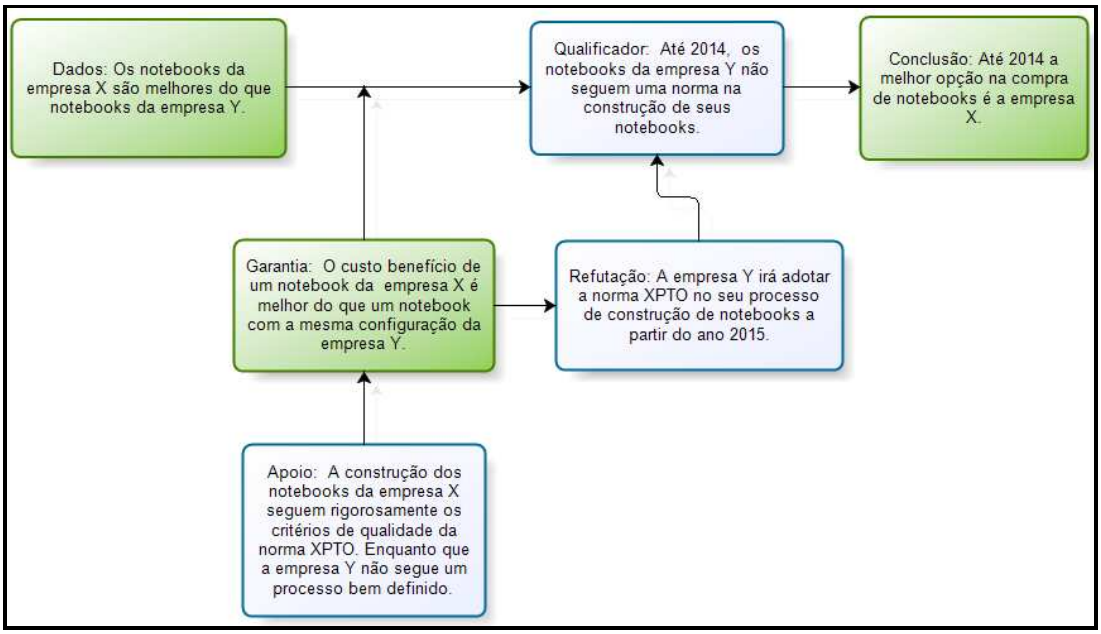

Figura 2 – Aplicação do Modelo de Toulmin.

Bentahar, Moulin e Bélanger (2010) definiram um conjunto de vantagens e desvantagens existentes no modelo de Toulmin e suas extensões. Apresenta-se na Tabela 3 o resultado da análise realizada pelos autores, bem como percepções obtidas através da construção do exemplo apresentado na Figura 2.

Tabela 3 – Avaliação do Modelo de Toulmin.

Vantagens	**Desvantagens**
Considera diversos componentes de um argumento e os relacionamentos entre eles.	Relação entre os elementos da argumentação pode ser ambígua. O modelo é baseado em lógica informal. As regras de inferência podem não estar claras, permitindo argumentos duvidosos.
Permite a suposição de regras de inferência para deduzir conclusões acerca das premissas identificadas.	É complicado relacionar os diversos elementos de um argumento em um processo de argumentação.
Baseia-se em pilares filosóficos e empíricos.	O critério de aceitabilidade dos argumentos não é especificado.

Continua na Próxima Página

Tabela 3 – *Continuação da Página Anterior*

Vantagens	**Desvantagens**
Facilita a construção de argumentos textuais.	Tem como foco a estruturação dos argumentos. Os participantes da discussão e suas bases de conhecimento não são considerados.
Oferece suporte a elicitação de conhecimento.	
Provê uma forma interessante de representar o conhecimento.	

2.2.2 Framework IBIS

O IBIS [3] destina-se a coordenação e planejamento dos processos de tomada de decisões políticas. Este framework tem como objetivo estimular o raciocínio investigatório que facilita a identificação de argumentos. Ele auxilia na elaboração de questões a serem debatidas, na exploração de posições que responde estas questões e no suporte do processo de disputa entre os envolvido.

Como um modelo de estruturação de discursos, o IBIS auxilia o processo de comunicação em um domínio de resolução de problemas. Isso permite que o sistema capture diferentes aspectos do problema com base em diferentes pontos de vistas dos participantes da discussão. Assumindo diversos pontos de vistas para um determinado problema, aumenta a chance de aplicar uma solução consistente para resolvê-lo (EBADI; PURVIS; PURVIS, 2009). Os elementos fundamentais do framework IBIS são descritos na Tabela 4.

Tabela 4 – Elementos do Framework IBIS.

Elemento	**Descrição**
Questão	Representa o objeto alvo de discussão. Possui o formato de questões e tem origem acerca de temas controversos.
Posição	Representa uma resposta que soluciona uma questão. Espera-se que uma questão tenha várias posições.
Argumento	Representa uma relação de apoio ou oposição a determinada posição. A validade de uma posição é definida a partir dos argumentos a favor e contra.

[3] Issue-Based Information System ou Sistemas de Informação Baseados em Questões

O IBIS framework tem como meta a conexão das questões chaves de um problema. Cada questão pode ter diversas posições. Uma posição é uma afirmação ou asserção que resolve a questão. Geralmente as posições são independente uma das outras. Os argumentos são utilizados para oferecer suporte ou objeção às posições. Apresentam-se na Figura 3 os possíveis relacionamentos entre os elementos do IBIS.

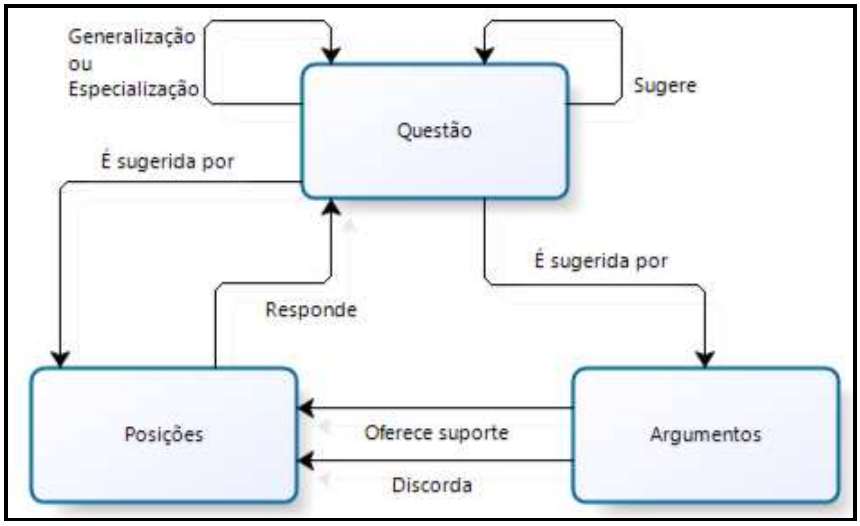

Figura 3 – Elementos do Framework IBIS.

Adaptado de (EBADI; PURVIS; PURVIS, 2009)

Com base na Figura 3 é possível observar a flexibilidade do IBIS. As questões podem ser geradas a partir das informações presentes em posições, argumentos e, até mesmo, em outras questões. Vale destacar a representação macro do argumento, contrapondo a representação interna proposta pelo modelo de Toulmin. Apresenta-se na Figura 4 um exemplo da aplicação do IBIS. Neste exemplo serão descritos alguns aspectos estruturais exibindo as características fundamentais deste framework.

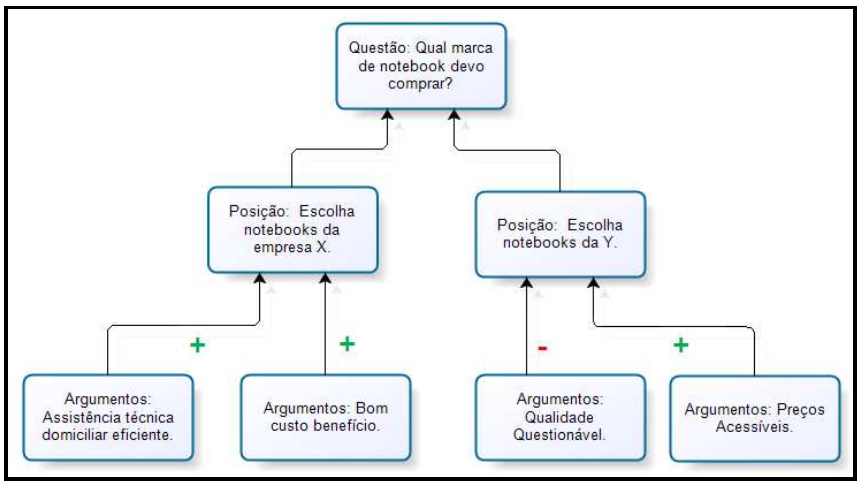

Figura 4 – Aplicação do Framework IBIS.

O exemplo apresentado na Figura 4 descreve uma discussão para determinar qual marca de notebook um participante deve comprar. O objeto alvo de discussão é interpretado como uma questão. Os pontos de vistas dos participantes da discussão são convertidos para as estruturas genéricas chamadas posições. Para finalizar, argumentos a favor ou contra cada posição são estabelecidos. Ao término do modelo, as decisões podem ser feitas com base em critérios bem definidos e o diagrama resultante pode ser documentado para facilitar a rastreabilidade das decisões tomadas.

A partir dos trabalhos de Ebadi, Purvis e Purvis (2009), Guerrero e Pino (2002), Conklin e Begeman (1998) e Kunz et al. (1970), uma análise envolvendo as vantagens e desvantagens do IBIS foi realizada. O resultado desta análise é apresentado na Tabela 5.

Tabela 5 – Avaliação do Framework IBIS.

Vantagens	**Desvantagens**
Permite a captura de diferentes aspectos do problema tendo como base pontos de vistas diferentes. Aumenta a probabilidade de seleção da melhor solução para uma questão específica.	Os participantes leigos neste framework têm dificuldades para classificar se um determinado conceito é uma questão, posição ou argumento.
Os três elementos básicos do modelo permitem a representação de qualquer atividade deliberativa.	Regras rígidas para a construção de um diagrama. Um argumento não pode ser o vértice inicial de uma discussão.
Aumenta a qualidade do diálogo, pois os participantes estão focados nas questões definidas.	Precisa de regras semânticas para facilitar a realização de inferências. Não possui um ponto de parada que represente o fim da discussão.
Aumenta a transparência do processo de tomadas de decisões. Os participantes podem rastrear os motivos que levaram uma determinada decisão.	A análise de diagramas contendo uma discussão extensa não é trivial. O grafo resultante pode dificultar a análise da argumentação.

2.2.3 Framework de Argumentação Abstrato

Em 1995, P. M. Dung apresentou um framework de argumentação abstrato (DUNG, 1995) com uma série de regras semânticas para avaliar a aceitabilidade dos argumentos. Em resumo, o framework proposto por Dung pode ser visualizado como um grafo direcionado contendo vértices que representam os argumentos e arestas que representam as relações binárias de ataque entre os argumentos (BREWKA; POLBERG; WOLTRAN, 2014). Os argumentos são entidades abstratas e não possuem uma estrutura interna em particular.

O framework de Dung é definido formalmente pelo par *F = (A, R)*. Onde que o A representa o conjunto de elementos abstratos chamados argumentos e o R representa as relações binárias entre estes argumentos. A seguir será realizado um exemplo demonstrando estes conceitos. Considere a seguinte argumentação a respeito da escolha de um notebook:

- A1: O notebook da empresa X possui desempenho superior a um notebook da empresa Y com a mesma configuração. Assim, certamente é o melhor custo benefício.
- A2: O notebook da empresa Y é mais barato do que o seu concorrente na empresa X. Assim, apesar de possuir desempenho menor, é o melhor custo benefício.
- A3: A qualidade das peças utilizadas na construção do notebook da empresa X são superiores as peças utilizados na construção dos notebooks da empresa Y. A vida útil do notebook da empresa X será maior do que o do concorrente.
- A4: Apesar de barato, vários consumidores estão reclamando do notebook da empresa Y devido a problemas de aquecimento.

Apresenta-se na Figura 5 a argumentação acerca da escolha de um notebook seguindo as diretrizes do framework de Dung. As setas representam as relações de ataque. Nesta discussão, *A = A1, A2, A3, A4* e *R = (A1 > A2), (A2 > A1), (A3 > A2), (A4 > A2)*. Vale ressaltar que um argumento que não foi atacado está aceito. Caso contrário, estará derrotado se nenhum outro argumento da discussão estiver defendendo o mesmo. O argumento que está atacando é válido apenas se estiver aceito. Caso dois argumentos aceitos ataquem um ao outro, uma situação de indefinição irá ocorrer. Para evitar esta situação, mais informações significantes devem ser adicionadas ao grafo da discussão.

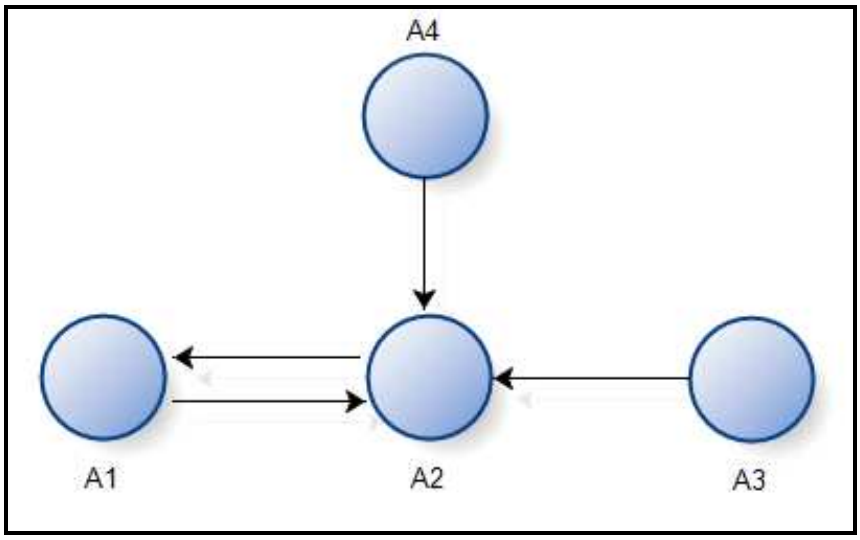

Figura 5 – Aplicação do Framework de Dung.

A partir do exemplo definido na Figura 5, dois conceitos fundamentais relacionados à aceitabilidade dos argumentos podem ser aplicados: O **conjunto livre de conflitos** representa um agrupamento de argumentos que não se atacam; **O conjunto admissível** é um conjunto livre de conflitos que defende-se de ataques de argumentos pertencentes a outras extensões. O conjunto $S = A1, A3, A4$ é um conjunto admissível livre de conflitos.

As semânticas utilizadas para verificar a aceitabilidade dos argumentos possibilitam diferentes formas de raciocínio. Elas produzem subconjuntos de argumentos válidos chamados extensões. Uma argumentação pode possuir várias extensões. Apresenta-se na Tabela 6 semânticas definidas no framework.

Tabela 6 – Semânticas do Framework de Dung.

Semântica	**Descrição**
Crédula (Credulous)	A maior extensão de argumentos válidos é selecionada. A extensão resultante é chamada de Extensão Preferida (Preferred Extension).
Cética (Sceptical)	A menor extensão de argumentos válidos é selecionada. A extensão resultante é chamada de Extensão Fundamentar (Grounded Extension).
Estável (Stable)	Uma extensão é dita estável caso ele derrote qualquer argumento que não pertença a ele. A extensão resultante é chamada de Extensão Estável (Stable Extension).

Aplicando as semânticas disponíveis no framework ao exemplo da Figura 5 obtém-se sempre o mesmo resultado. A discussão possui apenas uma extensão de argumentos admissível. Caso a argumentação possuísse várias extensões admissíveis, a análise ficaria da seguinte forma: Aplicando a semântica crédula resultaria no conjunto com o número máximo de argumentos aceitos; Aplicando a semântica cética resultaria no conjunto com o número mínimo de argumentos aceitos; Aplicando a semântica estável resultaria em todos os conjuntos que atacam todos os argumentos externos que não pertencem a ele.

A partir dos trabalhos de Dung (1995), Modgil e Prakken (2013), Nielsen e Parsons (2007), Zhang e Liang (2012), Prakken (2009) e Brewka, Polberg e Woltran (2014), uma análise envolvendo as vantagens e desvantagens do framework foi elaborada. Apresenta-se na Tabela 8 o resultado desta análise.

Tabela 7 – Avaliação do Framework de Dung.

Vantagens	**Desvantagens**
Notação simples e intuitiva. Estruturar a discussão em um grafo é uma tarefa trivial. Pois os argumentos são representados como vértices e existe apenas um relacionamento entre eles, a relação de ataque.	Relação de preferência entre argumentos em conflito não foi definida.
A aceitabilidade dos argumentos pode ser interpretada a partir de um ponto de vista cético, crédulo ou estável.	A relação de ataque entre argumentos é binária. Grupos de argumentos não podem realizar ou sofrer ataques.
Analisa a aceitabilidade dos argumentos de uma forma abstrata. Não considera estrutura interna, origem ou concepção dos argumentos.	
Semântica formal permitindo a adoção da abordagem em sistemas computacionais.	

2.2.4 Framework ACE

O ACE [4] (JURETA; MYLOPOULOS; FAULKNER, 2009) é um framework de argumentação baseado em proposições que possui origem na engenharia de requisitos. Este framework oferece maneiras de modelar e raciocinar acerca da validação relativa dos requisitos discutidos em uma reunião. A validação relativa pode ser vista como um consenso dos participantes de uma discussão a respeito de um artefato. O framework é composto de uma linguagem para representar as informações extraídas de uma discussão, uma condição de aceitabilidade para verificar a existência de consenso entre os participantes e procedimentos para checar a condição de aceitabilidade automaticamente.

Os modelos de argumentação, gerados com base na linguagem contida no ACE, são grafos direcionados com rótulos. Os vértices são classificados com base em quatro rótulos: i, It, P e C. Os vértices com rótulo (i) representam vértices de informação que servem de entrada ou saída para inferências (It). Os vértices com o rótulo (I) representam a aplicação de inferências a fim de obter determinadas saídas. Os vértices com rótulo (C) representam regras de conflito envolvendo dois ou mais vértices em um grafo. Finalmente, os vértices com rótulo (P) representam regras de preferência envolvendo a predileção de dois ou mais vértices do grafo. As arestas do grafo possuem apenas um rótulo (To).

[4] Acceptability Evaluation ou Avaliação da Aceitabilidade

Apresenta-se na Figura 6 um exemplo da aplicação da linguagem ACE. O exemplo modela uma discussão envolvendo uma provável aquisição de notebooks por parte de uma empresa. O grafo da discussão contem três proposições i(p1), i(p2) e i(p3) descritas a seguir:

- i(p1): Comprar um notebook da marca X oferece um bom custo benefício. Isto é essencial para empresa que esta enfrentando dificuldades financeiras.
- i(p2): O notebook da marca nacional X possui preço acessível e boa qualidade em suas peças.
- i(p3): Alguns notebooks nacionais da marca X apresentam defeitos após um ano de uso. É interessante adquirir notebooks da marca internacional Y. Apesar de mais caros, a qualidade do produto compensa a diferença de preços.

A proposição i(p2) foi inferida a partir de uma regra indutiva tendo como insumo i(p1). Com base nas ideias expressas em i(p1) pode-se inferir que o "bom custo-benefício representa um produto com qualidade e acessível no contexto da empresa que está enfrentando problemas financeiros. Porém, a proposição i(p3) contrapõe as ideias apresentadas em i(p1) exibindo argumentos contra a aquisição de notebooks de marcas nacionais. A relação de inferência entre as proposições i(p1) e i(p2) é descrita formalmente como *It(i(p1),i(p2))*. Enquanto que a relação de conflito envolvendo i(p3) e i(p2) é definida formalmente como *C1(i(p3),i(p2))*. Para finalizar, supõe-se que um questionário tenha sido aplicado na empresa e os funcionários preferiram os notebooks nacionais da marca X aos importados da marca Y. O vértice P1 representa uma regra de preferência entre as proposições i(p2) e i(p3). O uso da preferência gera um conflito *C2(P1, C1,i(p3))* que protege a proposição i(p3). A preferência é definida formalmente como *P1(i(p2),i(p3))*.

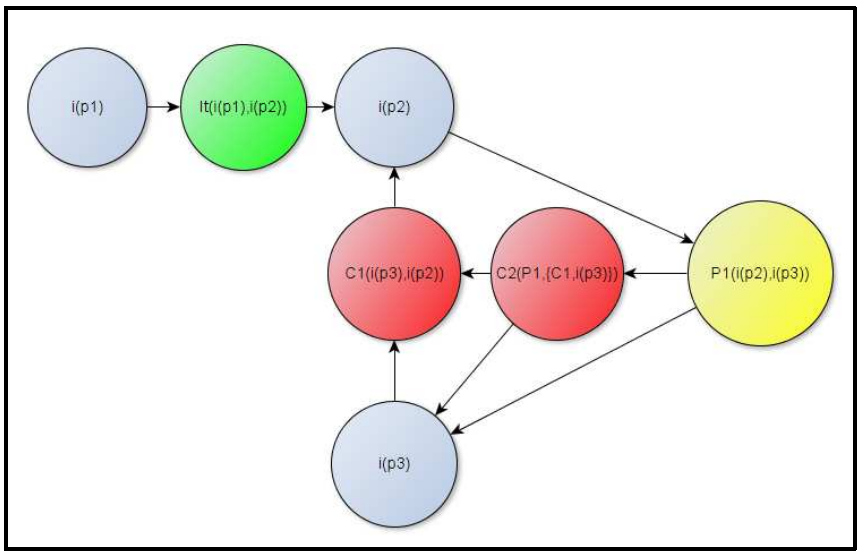

Figura 6 – Aplicação do Framework ACE.

A aceitabilidade proposta pelo ACE é baseada em três rótulos: A, AD e R. O rótulo A (Accepted) determina que um vértice do grafo foi aceito. O rótulo AD (Accepted and Dominated) é utilizado para representar vértices aceitos, porém, em uma relação de preferência, não foram priorizados. Finalmente, o rótulo R (Rejected) representa vértices que foram rejeitados por intermédio de um conflito. Com base nestes rótulos, cada vértice do grafo é computado individualmente.

Para definir a aceitabilidade de um vértice são executadas duas tarefas fundamentais: recuperação e avaliação. A tarefa de recuperação oferece uma rotina para identificar sub grafos com informações relevantes para a elaboração da condição de aceitabilidade de um vértice. O algoritmo de recuperação é baseado em uma busca em largura (BFS). Através da aplicação deste algoritmo é possível identificar todos os caminhos do grafo que terminam no vértice que está sendo avaliado. Já a tarefa de avaliação é mais complexa. segue uma descrição do processo de execução desta tarefa:

- Identificar os componentes fortemente conectados (SCC) do grafo de discussão;
- Definir o ordenamento topológico dos componentes fortemente conectados. Resultando em um grafo direcionado acíclico (DAG).
- Computar os rótulos A, AD e R para cada SCC do grafo. Para essa rotina existem duas abordagens:
  - SCC sem ciclos: Primeiramente, deve-se rotular tendo como base as posições definidas na ordenação topológica. Logo em seguida, o SCC que não contenha linhas de encontro é selecionado e rotulado. Os componentes já rotulados são ignorados, permitindo que o ciclo de rotulação continue nos outros componentes do DAG.
  - SCC com ciclos: Primeiramente, deve-se realizar a contagem n de todos os ciclos existentes no SCC. Logo após, todos os vértices do ciclo recebem o rótulo A. A partir do último vértice adicionado ao SCC, n agentes irão percorrer e rotular os vértices de todos os caminhos possíveis até retornarem ao vértice inicial e rotulá-lo em conjunto. Uma sequência de rótulos de avaliação é gerada para cada vértice. O critério de parada é atingindo quando os dois últimos rótulos do vértice inicial são iguais. Nesta situação, a aceitabilidade de cada vértice é definida como sendo o último elemento da sequência de rótulos. Caso essa condição não seja satisfeita, os agentes serão enviados novamente até que a sequência do vértice inicial possua quatro rótulos. Persistindo a diferença entre os dois últimos rótulos do vértice inicial, a aceitabilidade se torna inconclusiva. A adição de informações relevantes será necessária para a avaliação da aceitabilidade.

Apresenta-se na Figura 7 a avaliação da aceitabilidade do vértice i(p2) a partir da discussão definida na Figura 6.

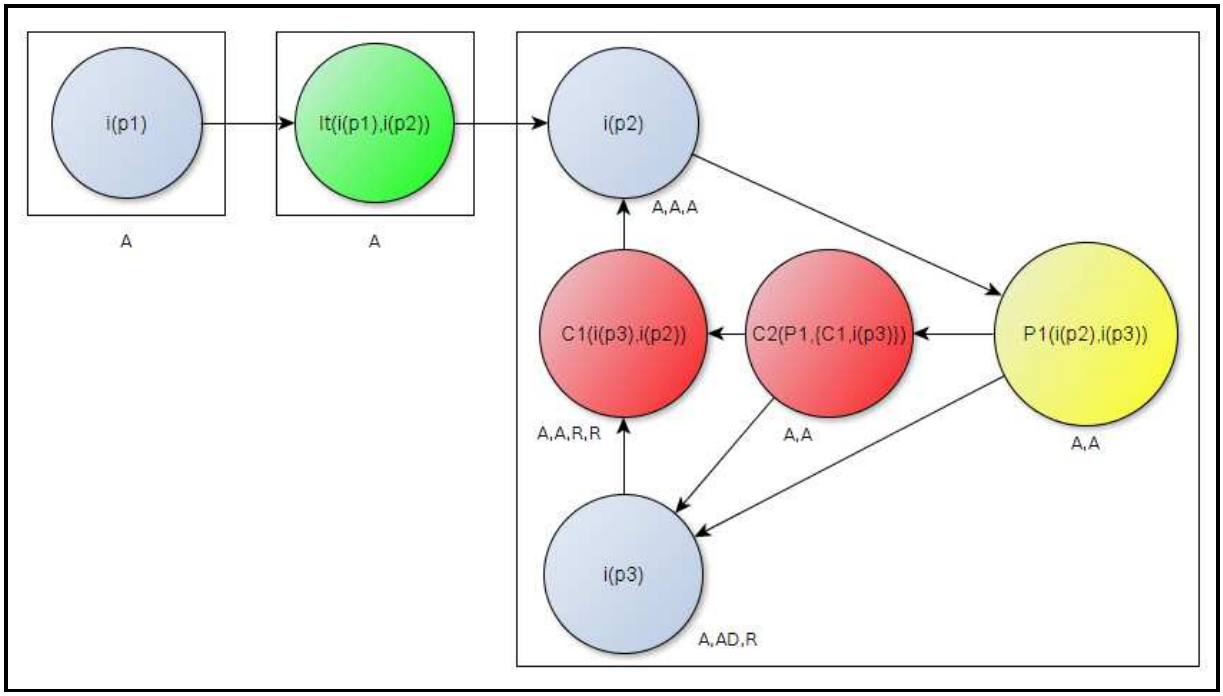

Figura 7 – Avaliação da Aceitabilidade do Framework ACE.

Ao término da avaliação da aceitabilidade do vértice i(p2), observa-se que todos os vértices relacionados também foram avaliados. Apenas o conflito C1 e a proposição i(p3) foram rejeitados. As regras bem como os algoritmos utilizados para computar os rótulos A, AD e R nos vértices de um grafo estão disponíveis em anexo.

Com base no trabalho de (JURETA; MYLOPOULOS; FAULKNER, 2009) e nos exemplos desenvolvidos na Figura 6 e 7, apresenta-se na Tabela **??** as vantagens e desvantagens do framework ACE.

Tabela 8 – Avaliação do Framework ACE.

Vantagens	**Desvantagens**
Define uma condição de aceitabilidade que determina a validação relativa de um artefato.	Usuários leigos no contexto da argumentação podem ter problemas para entender os relacionamentos entre os elementos do grafo.

Continua na Próxima Página

Tabela 8 – *Continuação da Página Anterior*

Vantagens	**Desvantagens**
A linguagem oferecida é simples e expressiva. Facilitando a representação dos elementos presentes em uma discussão.	Não foi encontrada nenhuma ferramenta computacional que se baseie neste framework.
Provê algoritmos para a recuperação e avaliação dos argumentos de uma discussão.	

2.3 Argumentação na Web

Diversas áreas de pesquisa tais como inteligência artificial, filosofia, linguística e engenharia de software estão propondo diferentes abordagens que compreendem técnicas de argumentação. O interesse na argumentação é evidente, porém as diferentes abordagens desenvolvidas consideram aspectos específicos do domínio de aplicação. Isto acaba dificultando a integração e unificação dos resultados de cada proposta em um modelo de argumentação geral e coerente (BEX; PRAKKEN; REED, 2010). Estas várias abordagens dificultam a integração dos dados gerados a partir das ferramentas computacionais que auxiliam o processo de argumentação e geralmente não ofertam técnicas para lidar com a lógica da argumentação (RAHWAN et al., 2011).

Para solucionar os problemas acima, uma linguagem de marcação chamada AIF[5] (RAHWAN et al., 2011) foi elaborada. O AIF consiste de uma ontologia que unifica conceitos semelhantes que são utilizados na argumentação em vários contextos. Os principais objetivos do AIF são: facilitar a troca de informações entre ferramentas que suportam o processo de argumentação; simplificar o desenvolvimento de sistemas multiagentes, principalmente na racionalidade dos agentes inteligentes.

O AIF pode ser dividida em duas ontologias: superior e formal. A ontologia superior define uma linguagem contendo diversos elementos para representar uma discussão. A ontologia formal oferece uma série de padrões que auxiliam a análise racional dos argumentos de um discurso. Os padrões podem variar dependendo do contexto que ocorre o diálogo. Apresenta-se na Figura 8 a estrutura geral e os relacionamentos dos conceitos presentes no AIF.

[5] Argument Interchange Format ou Formato para a Troca de Argumentos

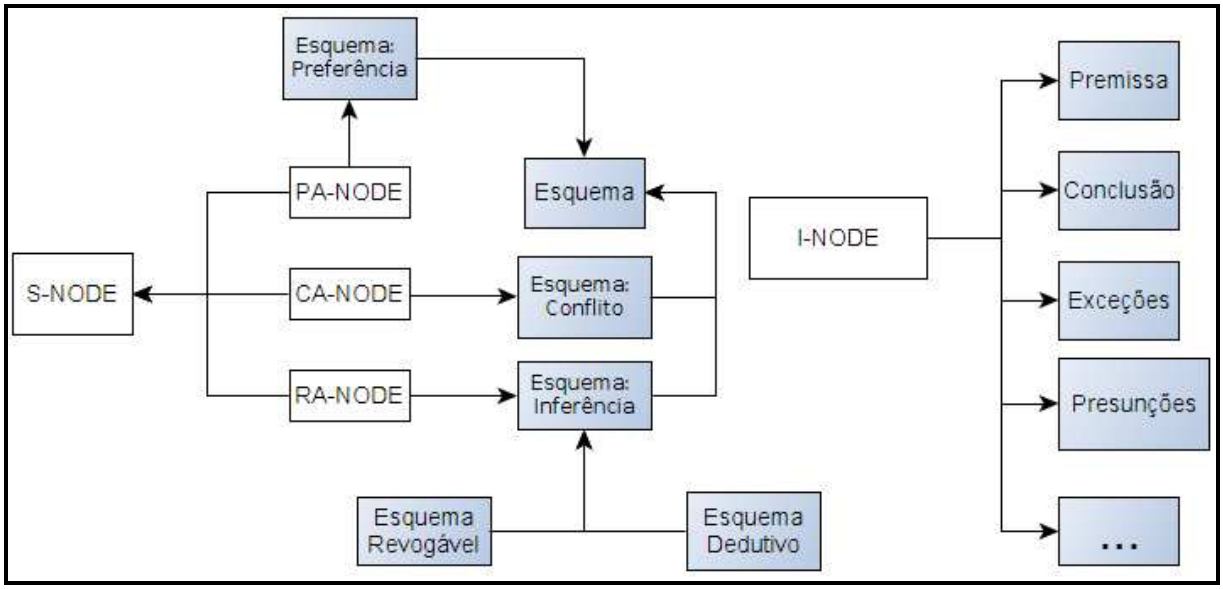

Figura 8 – Estrutura da Ontologia AIF.

Adaptado de (BEX; PRAKKEN; REED, 2010)

Com base na Figura 8, observa-se que a ontologia define quatro tipos de nó. Os nós *I-NODE*[6] encapsulam as informações disponíveis na discussão. Estas informações podem possuir caráter dedutível: premissas ou conclusões, ou revogável: presunções ou exceções. Os nós de informação representam a base de conhecimento concebida como conseqüência de uma discussão. Os nós *S-NODE*[7] representam aplicações de esquemas, ou seja, padrões racionais. A partir deles, regras de inferência, preferência e conflito podem ser estabelecidas. Os esquemas de argumentação podem ser classificados. O esquema de inferência foi classificado em duas diretrizes diferentes: *Dedutivo* e *Revogável*. O esquema dedutivo compreende aspectos relacionados a lógica formal. Esta abordagem consiste no processo de gerar conclusões a partir de premissas válidas de forma monotônica. Já o esquema revogável considera a lógica não monotônica, ou seja, a adição de novas informações, confiáveis ou não, pode modificar o resultado de uma ou várias conclusões.

Apresenta-se na Figura 9 um exemplo de aplicação do AIF. O exemplo consiste de uma discussão envolvendo a venda de um notebook. Os nós de informação e de esquema são representados, respectivamente, pela cor azul e laranja. Os nós de esquema são gerados a partir de percepções lógicas baseadas nas informações presentes na discussão. As informações são apresentados a seguir:

- *I1* João garante para José que o notebook está sem problemas.
- *I2* O notebook está sem problemas.

[6] Information Node ou Nó de Informação
[7] Scheme Node ou Nó de Esquema

- *I3* O notebook está funcionando corretamente.
- *I4* José ouviu falar que João não é confiável.
- *I5* João afirma que é confiável.
- *PA1* Por meio de uma pesquisa com seus amigos, José descobriu que João é confiável.

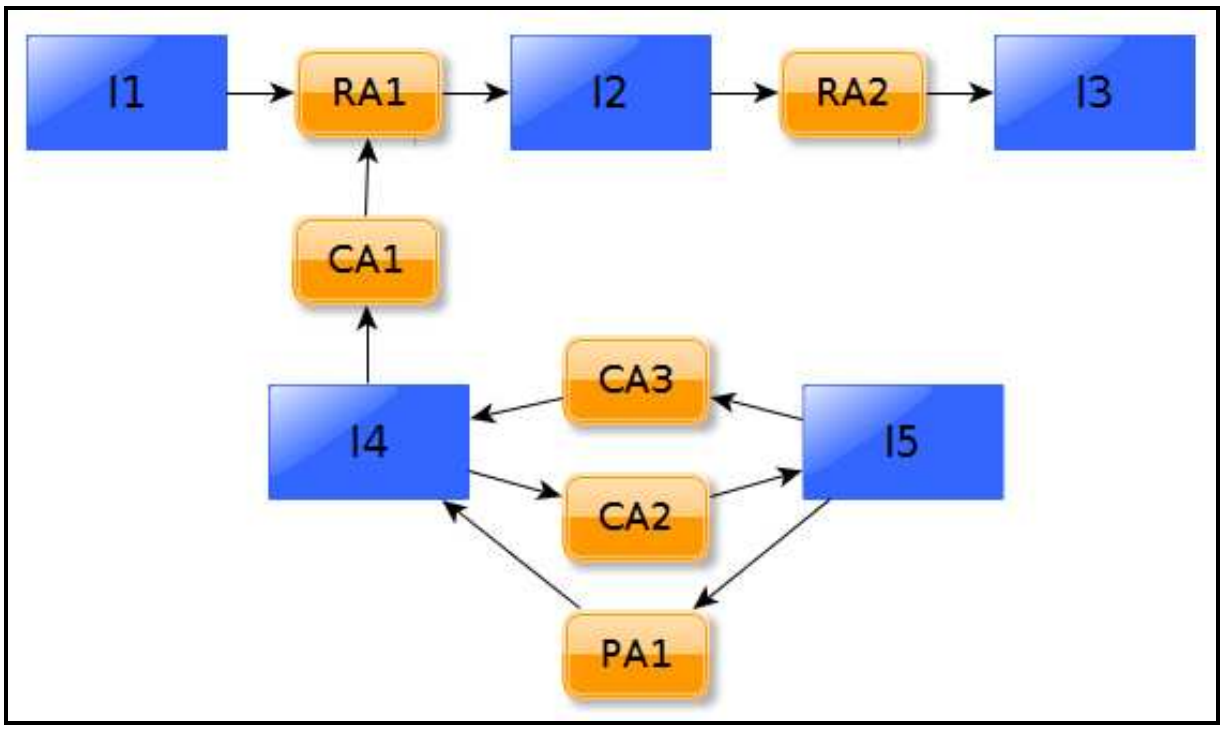

Figura 9 – Aplicação da Ontologia AIF.

O nó de informação *I2* foi gerado através de uma regra de inferência *RA1* revogável, as premissas que oferecem suporte para I2 possuem certas restrições. O conflito *CA1*, gerado a partir do nó *I4*, está atacando *RA1*. A conclusão *I2* é válida por conta da aplicação de uma regra de preferência *PA1* que apoia o nó *I5*, contrário ao nó *I4*. Para finalizar, uma conclusão pode servir de insumo para uma nova regra de inferência. O nó *I3* foi gerado por intermédio da regra de inferência dedutiva *RA2*, que teve como insumo a conclusão *I2*.

3 Metodologias e Tecnologias

Neste capítulo, apresentam-se as metodologias e tecnologias utilizados para a realização deste trabalho. Para isto, demostra-se cada abordagem, seja ela aplicada na gerência do projeto ou no desenvolvimento da ferramenta de argumentação.

3.1 Metodologia

Apresenta-se nesta seção as metodologias utilizadas no gerenciamento e desenvolvimento deste trabalho.

3.1.1 Scrum

Desde o início dos anos 90, o Scrum vem sendo utilizado para gerenciar o desenvolvimento de produtos complexos. Segundo (SCHWABER; SUTHERLAND, 2013) o Scrum é um framework estrutural que, a partir de regras bem definidas, permite a construção de produtos com alto valor de uma maneira produtiva e criativa. As regras do Scrum são dispostas em um processo iterativo e incremental. Apresenta-se na Figura 10 a estrutura geral do Scrum.

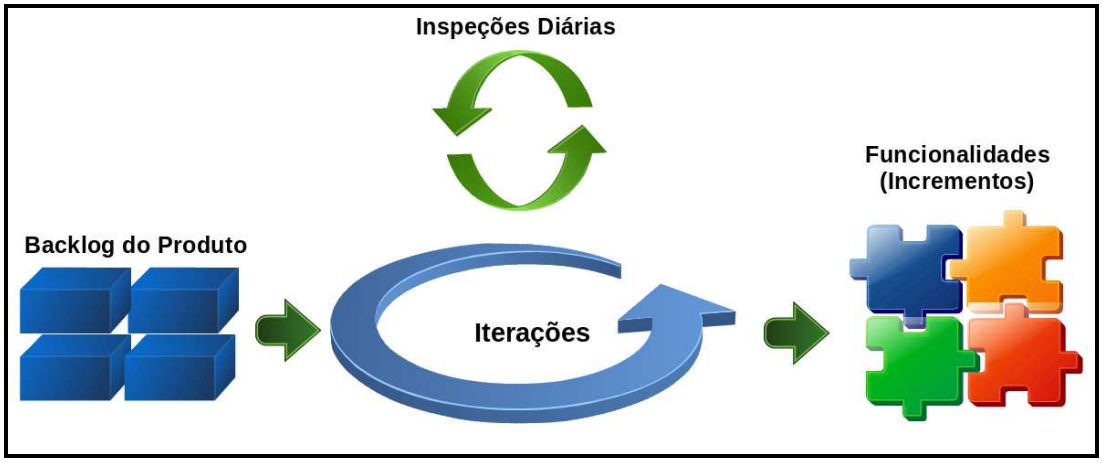

Figura 10 – Estrutura do Framework Scrum.

Adaptado de (SCHWABER, 2004)

A estrutura apresentada na Figura 10 funciona da seguinte forma. O laço inferior representa uma iteração contendo atividades de desenvolvimento. Esta iteração, também conhecida com *Sprint*, tem como entrada os requisitos presentes no *backlog* do produto. Durante a iteração, aconselha-se a ocorrência de reuniões diárias para que cada indivíduo

da equipe possa inspecionar as atividades de seus companheiros. Ao término da iteração, um incremento do produto final é desenvolvido. Permitindo aos usuários chaves a chance de avaliar a funcionalidade e sugerir melhorias para o projeto.

Os times do Scrum são entidades auto-organizáveis e multifuncionais. A equipe deve escolher qual a melhor maneira de realizar seu trabalho. Para concluir suas pendências, exigi-se da equipe as competências necessárias sem depender de terceiros que não participam da equipe. O time é dividido da seguinte forma:

- *Product Owner*: Possui conhecimento acerca do contexto da aplicação. Seu principal papel é agregar valor ao negócio. Para isto, realiza o gerenciamento do *backlog* do produto, adicionando e priorizando requisitos.
- Equipe de Desenvolvimento: Profissionais que implementam os requisitos contidos no *backlog* do produto. Seu principal papel é entregar uma versão funcional do produto.
- *Scrum Master*: Remove possíveis impedimentos que poderiam prejudicar a execução do projetos. Possui papel de facilitador, aplicando as regras do Scrum de forma correta.

O Scrum define uma série de eventos *time-boxed* com o objetivo de criar uma simples rotina a ser seguida. Os eventos possuem uma duração máxima, porém, caso o objetivo do evento tenha sido concluído, ele pode ser finalizado previamente. Apresenta-se na Tabela 9 a descrição dos principais eventos.

Tabela 9 – Eventos do Scrum.

Evento	**Descrição**
Iteração (*Sprint*)	Possui duração de aproximadamente um mês, em que uma versão funcional do produto é desenvolvida. Ao término de uma iteração, inicia-se outra. A iteração é composta de reunião de planejamento, encontros diários, implementação dos requisitos, revisão e retrospectiva da *Sprint*.

Continua na Próxima Página

Tabela 9 – *Continuação da Página Anterior*

Evento	**Descrição**
Reunião Diária	Possui duração de aproximadamente 15 minutos. O papel deste evento é permitir que a equipe avalie as atividades realizadas bem como a determinação de metas a serem concluídas até a próxima reunião diária.
Revisão da *Sprint*	Possui duração de aproximadamente 4 horas. Ao término da *Sprint*, o incremento desenvolvido deve ser inspecionado para verificar sua conformidade com o *backlog* do produto. Durante a reunião de revisão, sugestões para otimizar o valor do produto são realizadas.
Retrospectiva da *Sprint*	Possui duração de aproximadamente 3 horas. Este evento representa uma oportunidade para a equipe identificar melhorias a serem aplicadas em *Sprints* futuras. Para isto, é feita uma reflexão acerca dos pontos positivos e negativos da *Sprint* avaliada.

Além dos eventos, o Scrum contem alguns artefatos. O ***backlog* do produto** representa uma lista contendo todos os requisitos que devem ser implementado. Geralmente os requisitos são descritos por meio de histórias de usuários. A lista pode sofrer mudanças a qualquer momento e reflete a visão que o *Product Owner* possui do negócio. O ***backlog* da *Sprint*** é uma lista de itens que foram selecionados com base no *backlog* do produto. Esta lista representa o trabalho a ser feito na *Sprint*.

Apesar de possuir uma difícil implantação, o Scrum se mostrou extremamente eficaz quando adotado corretamente. (PHAM; PHAM, 2012) indica quatro vantagens obtidas com a adoção do Scrum:

- Mecanismos sistemáticos para a redução dos riscos: Através da inspeção freqüente e dos ciclos adaptativos o Scrum mitiga, com antecedência, possíveis inconsistências que poderiam dificultar a execução do projeto.

- Ciclo de desenvolvimento de software sucinto: Por meio de iterações curtas, a equipe de desenvolvimento implementa versões incrementais do produto. Cada iteração

dura aproximadamente um mês e pode compreender atividades de todo o processo de desenvolvimento de software.

- Processo de gerenciamento de projetos adaptativo: As mudanças são vistas como oportunidades para melhorar. Ao término de cada iteração, reuniões para identificar inconsistências são realizadas. As lições aprendidas são aplicadas na definição de novas iterações.

- Abordagem baseada na motivação e satisfação das pessoas: Seguindo os princípios da metodologia ágil, o objetivo é permitir que a equipe decida quais ações tomar para cumprir com a meta definida.

3.1.2 Engenharia de Requisitos: Orientação à Meta

Os sistemas de informação estão cada vez mais inclusos no cotidiano das pessoas. Um fator a ser considerado no desenvolvimento de software é o ambiente no qual ele operará. A partir desta informação, aspectos sociais baseados no público alvo, condições de uso e restrições do software podem ser identificados. Modelos de desenvolvimento de software tradicionais pecam na inclusão destes aspectos sociais. Isto acaba diminuindo a aceitação do software pelos *stackholders* (BORGIDA et al., 2009).

Com base em um ponto de vista social, é fácil perceber que o mundo é intencional, ou seja, seus comportamentos apresentam motivações subjacentes que o justificam. Atores intencionais possuem vontades e desejos. Para satisfazê-los, ações ou tarefas são realizadas. A escolha de quais ações ou tarefas executar fica a critério do ator (YU et al., 2011).

Segundo Lamsweerde (2001) a engenharia de requisitos orientada à meta "*preocupa-se com a utilização de metas para elicitar, elaborar, estruturar, especificar, analisar, negociar, documentar e modificar requisitos*". A partir das metas, é possível identificar vários objetivos que o sistema a ser desenvolvido deve alcançar. Para aproximar os conceitos de orientação à meta para a engenharia de requisitos, foi desenvolvido um framework conceitual I*[1] (I Star) (YU, 1996).

O framework I* oferece uma linguagem para especificar requisitos utilizando aspectos sociais. Para isto, dois tipos modelos estão disponíveis: modelo de dependência estratégica (*Strategic Dependency Model*) (SD) e modelo de raciocínio estratégico (*Strategic Rationale Model*) SR. O SD é utilizado para expressar as dependências entre os atores. Enquanto que o SR descreve a lógica racional por traz destas dependências.

Os elementos fundamentais do framework i* são: (i) atores - entidades ativas que executam atividades para alcançar metas; (ii) metas rígidas - representam os desejos intencionais, ou seja, os objetivos de um ator; (iii) metas flexíveis - são semelhantes às metas,

[1] Informações gerais disponíveis em: http://$istar.rwth-aachen.de/tiki-view_articles.php$

porém não possuem critérios de satisfação bem definidos. A aprovação deste elemento depende do ponto de vista do interessado; (iv) tarefas - representam as etapas realizadas por um ator visando satisfazer uma meta rígida. O framework I* também oferece elos entre os elementos, tais como meio-fim, contribui-positivamente, contribui-negativamente, decomposição, entre outros.

Modelar um ambiente considerando as perspectivas sociais implica em determinar os seguintes fatores:

- Identificar os atores relevantes do ambiente;
- Analisar as intenções, ou seja, as motivações por trás de cada ator;
- Identificar relações de dependência entre os atores;
- Montar uma rede de relacionamento entre os atores.

Apresenta-se na Figura 11 um exemplo de aplicação do framework I*. O exemplo consiste na realização da matrícula semestral em uma universidade e possui dois atores, aluno e secretaria. O ator aluno possui a meta “Matrícula escolar seja realizada”. Esta meta pode ser realizada de duas formas, “Realizar a matrícula pela internet” ou “Realizar a matrícula presencialmente”. Cada uma destas tarefas impacta positivamente ou negativamente nas metas flexíveis que foram definidas, “Agilidade” e “Suporte do Coordenador”. Estas metas flexíveis definem critérios de qualidade que auxiliam na escolha de uma tarefa. Caso o aluno queira fazer a matricula presencialmente, ele dependerá da secretária. O recurso “Ficha de Matrícula” é necessário para a realização da matrícula. O ator secretaria possui a meta “Matrícula do aluno seja efetivada”. Essa meta é alcançada por meio da tarefa “Realizar matrícula através de fichas”. Durante a execução desta tarefa o recurso “Ficha de Matrícula” é disponibilizado para o aluno. Para finalizar, a realização da matrícula através de fichas acaba impactando negativamente na meta flexível “Rapidez no atendimento”.

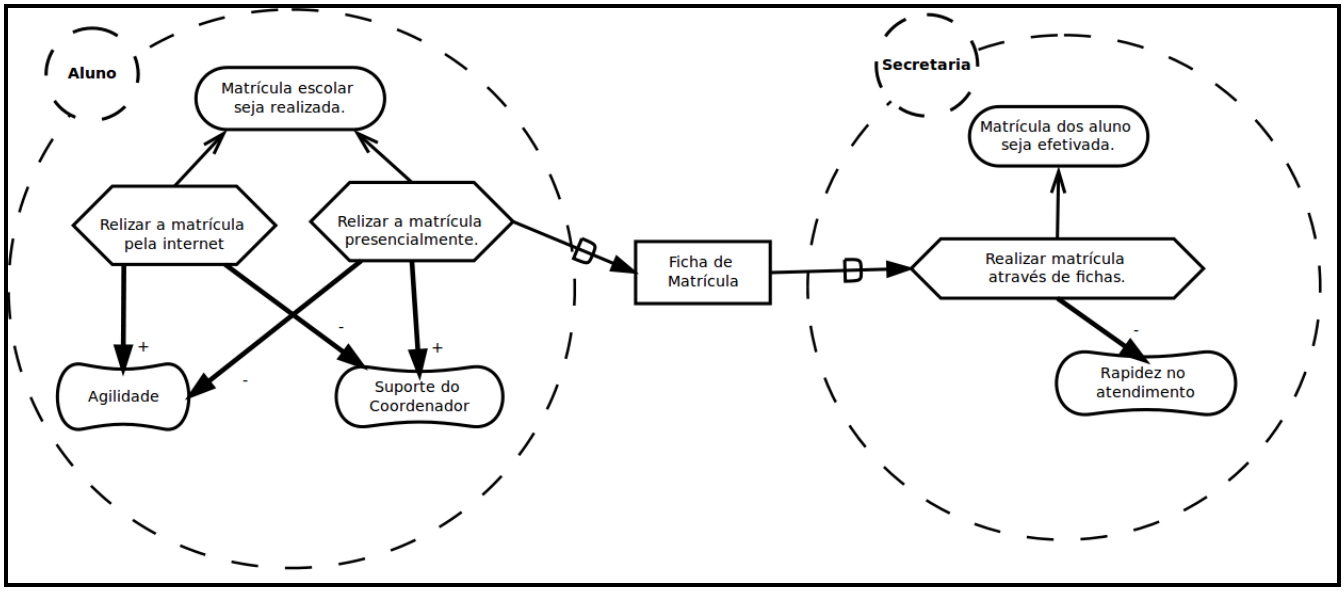

Figura 11 – Aplicação do Framework I*

A modelagem social é bastante flexível. Ela pode ser utilizada em diversos contextos, a metodologia ágil é um bom exemplo. Através de um estudo de caso, foi demonstrado que a utilização do framework I* enriquece a especificação das histórias de usuário (JAQUEIRA et al., 2013).

Utilizando a orientação a metas é possível determinar a origem dos conceitos existentes na especificação de requisitos. Para cada meta definida várias alternativas de solução devem ser determinadas. Ao longo da elicitação diversas validações são realizadas, assim requisitos inconsistentes expressadas pelo ator são identificados e removidos rapidamente (ZDRAVKOVIC; SVEE; GIANNOULIS, 2013).

3.2 Tecnologia

Apresenta-se nesta seção as tecnologias utilizadas no desenvolvimento da ferramenta de argumentação.

3.2.1 Grails: Groovy on Rails

O Grails é um framework para desenvolvimento de aplicações Web de alta produtividade, inspirado no Ruby on Rails, entretanto, utilizando linguagens como Java e Groovy. Este framework provê diferentes suportes e modelos visando maior produtividade, extensibilidade, reusabilidade, portabilidade, confiabilidade, dentre outros critérios de qualidade.

O framework Grails oferece suporte baseado em bibliotecas Java bem estabelecidas no mercado, tais como: Hibernate, Maven, SiteMesh, Spring, dentre outros. Adicionalmente, o Grails pode ser integrado a qualquer biblioteca Java através de plugins ou acesso direto. Todas as boas diretrizes do Java EE foram herdadas pelo Grails.

Vale ressaltar que, segundo (SMITH; LEDBROOK, 2009), o Grails faz parte da próxima geração de frameworks de desenvolvimento Web baseados em Java, oferecendo ao desenvolvedor a possibilidade de obter, dentre outras vantagens, uma alta produtividade e reusabilidade. O Grails pode ser estendido por meio de plugins. No site oficial[2] existem mais de mil opções, que oferecem serviços de segurança, teste, melhoria no desempenho, requisições AJAX, entre outros.

Para o aumento da produtividade, o Grails permite a codificação através do Groovy. O Groovy é uma linguagem dinâmica com sintaxe semelhante ao Java, compila para *bytecodes* e executa na JVM. Apesar de semelhante, a sintaxe do Groovy é mais flexível e poderosa do que a sintaxe do Java. Permitindo a construção, desde simples *shell scripts* a até aplicações robustas com milhares de linhas de código (ABDUL-JAWAD, 2009).

A estrutura oferecida pelo Grails procura ser simples e consistente no que se refere ao uso de frameworks e bibliotecas já consolidadas no mercado. O desenvolvedor pode-se concentrar nas regras de negócio e em aspectos específicos do domínio cognitivo de interesse, uma vez que detalhes técnicos de configuração, os quais demandam um tempo considerável de desenvolvimento em frameworks tradicionais, são altamente facilitados no ambiente de desenvolvimento do Grails.

No intuito de prover uma maior compreensão quanto a execução de uma aplicação em Grails, a Figura 12 ilustra os principais elementos envolvidos no funcionamento do núcleo da aplicação.

[2] *http://grails.org/plugins*

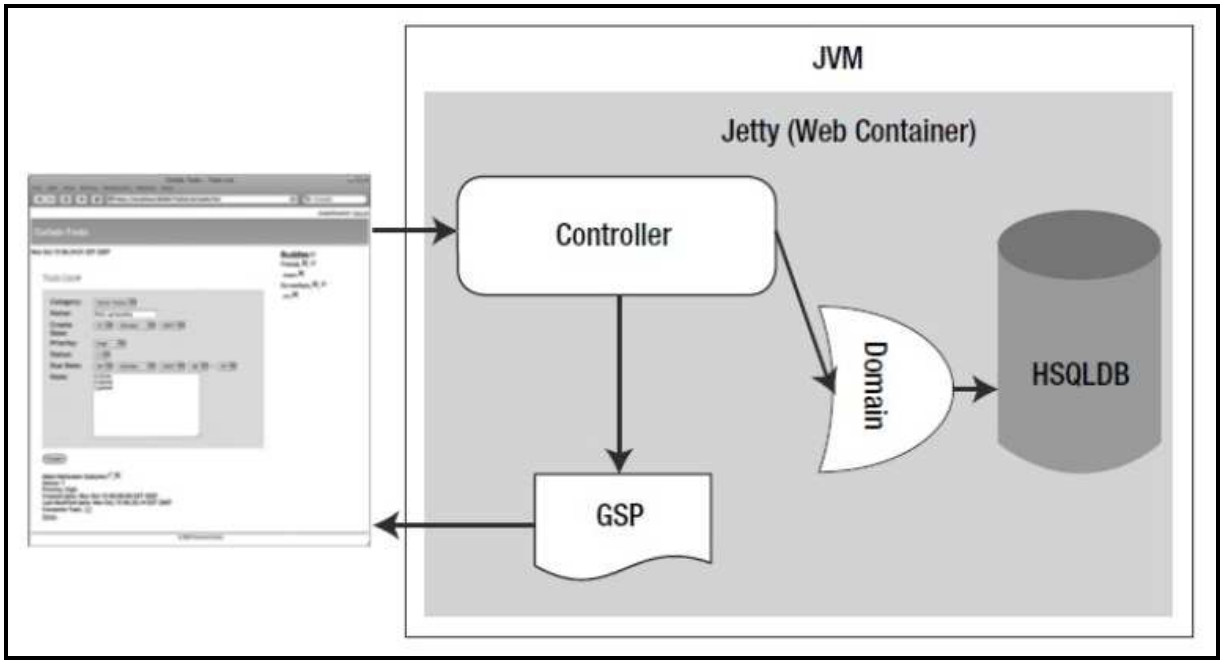

Figura 12 – Visão de Execução do Grails

Extraído de (JUDD; NUSAIRAT; SHINGLER, 2008)

Inicialmente, é realizada uma requisição através do browser para um servidor Web Jetty (i.e. Apache Tomcat ou outro). A controladora recebe e trata essa requisição podendo resolvê-la sozinha ou requisitando os serviços de alguma classe de domínio. Todas as classes contidas no domínio são persistidas utilizando o framework GORM. Assim, padrões como DAO ou *Repositor* não precisam ser utilizados. Quando a controladora finaliza a operação requisitada, ela envia o resultado para o GSP, que representa a view que irá renderizar o HTML, o qual será retornado para o browser que o havia requerido.

O Grails possui características semelhantes ao Ruby on Rails (ROR). Dada a complexidade dessa plataforma, vale destacar que o Grails, assim como o ROR, estabelece uma arquitetura própria, baseada, principalmente, nos componentes ilustrados na Figura 13.

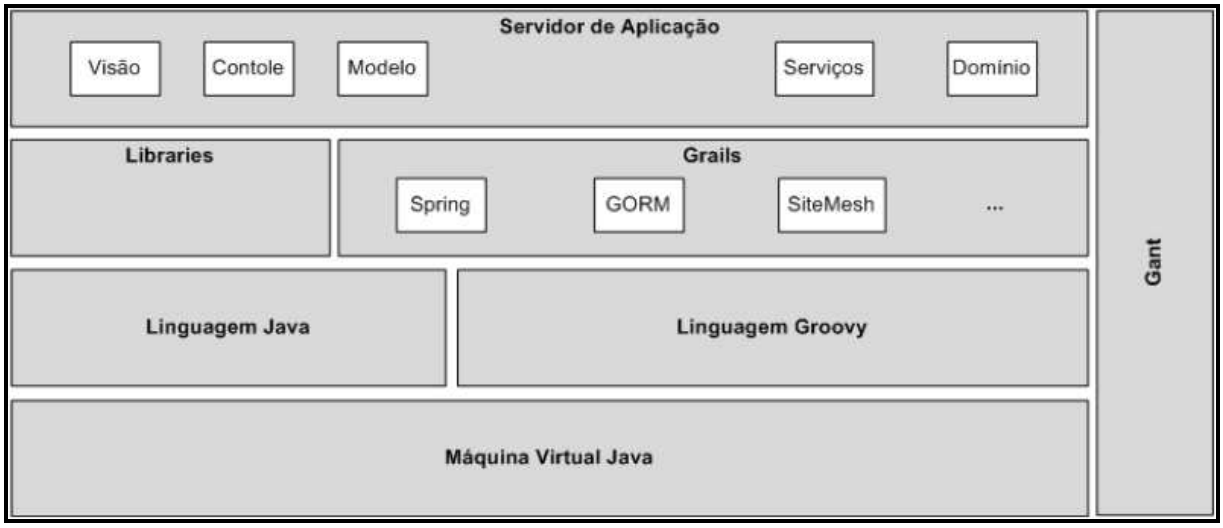

Figura 13 – Arquitetura do Grails

Adaptação de (JUDD; NUSAIRAT; SHINGLER, 2008)

Como apresentado, a base do Grails é a Máquina Virtual Java (JVM), programa que carrega e executa as aplicações Java, convertendo *bytecodes* em código executável, compreendidos pela máquina. Acima da JVM são vistas duas linguagens, a linguagem Java e a linguagem Groovy. A separação entre a linguagem Java e a JVM deve-se à constante criação de novas bibliotecas para a plataforma.

O framework Grails provê compatibilidade com qualquer tipo de código escrito em Java. Sendo assim, caso um software legado em Java precise ser reaproveitado, o Grails é capaz de realizar esta tarefa através dos seus recursos sem maiores problemas.

Na camada logo acima das linguagens, o núcleo do Grails é de fato encontrado, ele é composto de projetos *open-source*, desenvolvidos pela indústria (ex. o framework Spring), oferecendo um conjunto de componentes facilmente integráveis à aplicação. Adicionalmente, permite ao desenvolvedor concentrar-se apenas nas regras de negócio, cabendo, por exemplo: (i) ao SiteMesh a definição de layouts reutilizáveis para páginas GSP ou JSP, e (ii) ao GORM prover elementos para definir relacionamentos entre classes de domínio. Outras bibliotecas construídas a partir de Groovy ou Java podem ser utilizadas sem problemas, seja código aberto ou proprietário.

A última camada da arquitetura interna do Grails é a de aplicação. É nesta camada que será implementada toda a aplicação do sistema (Classes de Domínio, Controladoras, Visões, dentre outros detalhes). Ressalta-se o uso do padrão arquitetural *Model-View-Controller* (MVC), o qual facilita a organização e execução do sistema em desenvolvimento.

A fim de gerenciar os projetos e artefatos Grails, a arquitetura interna deste fra-

mework provê um componente chamado Gant. O Gant é um framework que gerencia *builds*, utilizando a linguagem Groovy na criação de Scripts Apache Ant.

3.2.2 Node.JS

O Node.js é uma plataforma desenvolvida a partir da *engine* JavaScript V8 da Google. Esta plataforma permite a construção de aplicações em rede escaláveis de forma trivial e produtiva. Para isto, oferece um modelo orientado a eventos assíncronos com um esquema de recursos *non-blocking*, que o faz uma solução leve e eficiente para aplicações em tempo real com intenso uso de dados. Diferente de vários plataformas *open-source*, o Node.js é fácil de inicializar e não requer recursos da máquina em excesso, como memória ou espaço de disco.

Atualmente, existem várias tecnologias consolidadas para *front-end* e *back-end*. O papel do Node.js é oferecer uma interface consistente entre estas duas tecnologias. Apresenta-se na Figura 14 um esquema exibindo onde o Node.js se encaixa no contexto de desenvolvimento de software.

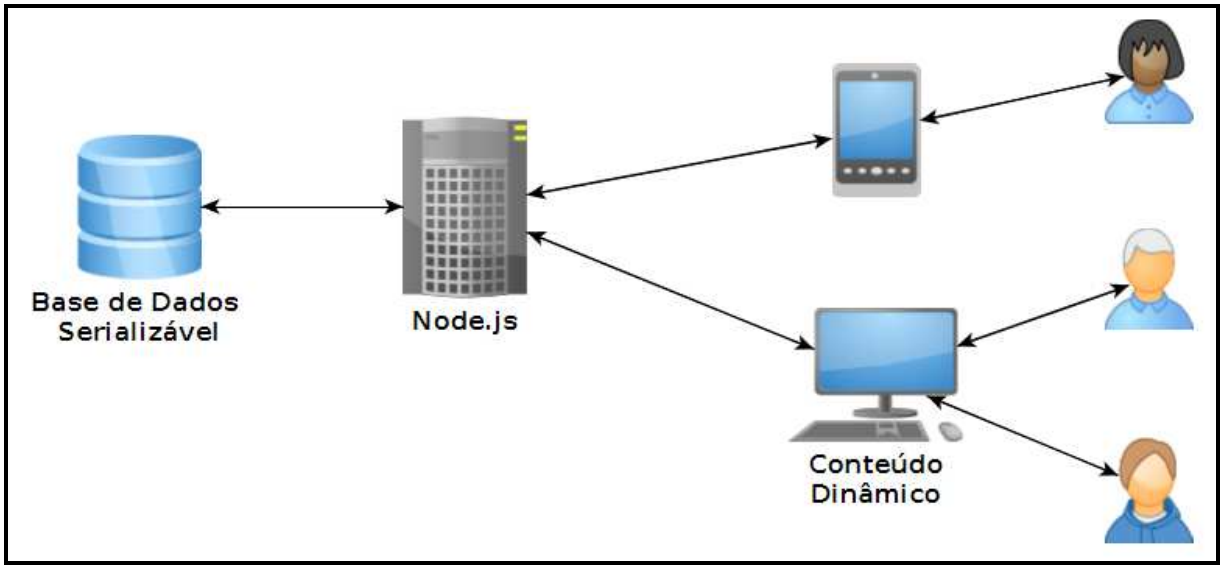

Figura 14 – Atuação do Node.js

Adaptação de (WILSON, 2013)

O Node.js exerce seu trabalho através de um laço de eventos. O laço funciona da seguinte forma: (i) Carrega e executa o programa; (ii) inicializa o laço de eventos; (iii) espera algum evento ser disparado; (iv) executa os manipuladores de eventos; (v) finaliza o processo, se o mesmo estiver ocioso. Sempre que um evento ocorrer, o Node.js irá executar as *callbacks* que estão escutando aquele evento. Apresenta-se na Figura 15 o laço de eventos do Node.js.

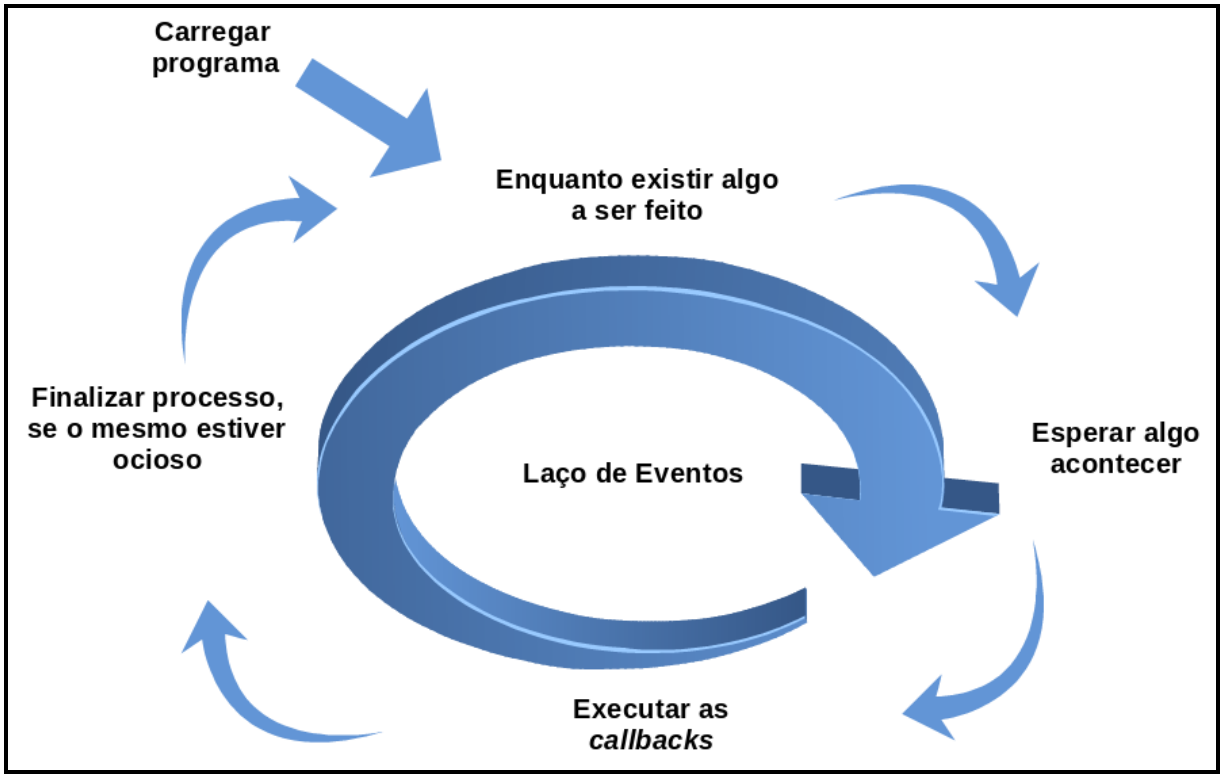

Figura 15 – Laço de Eventos do Node.js

Adaptação de (WILSON, 2013)

As plataformas tradicionais que oferecem serviços web exigem que para cada requisição, uma *thread* seja criada. Cada nova *thread* consome uma quantidade específica de memória RAM. Uma aplicação contendo um alto fluxo de requisições necessitaria de uma infraestrutura consistente para funcionar adequadamente. O Node.js funciona de uma forma diferente, ele opera em uma única *thread* utilizando chamadas de entrada e saída não bloqueantes. Assim, permitindo milhares de requisições concorrentes e poupando, consideravelmente, o consumo dos recursos computacionais.

3.2.3 Twitter Bootstrap

A partir da necessidade de padronizar as ferramentas e bibliotecas *front-end* da empresa Twitter, Mark Otto e Jacob Thornton desenvolveram um framework *open source* chamado Bootstrap. Este framework oferece um projeto baseado em arquivos CSS, plugins JavaScript e ícones que permitem o desenvolvimento de interfaces gráficas responsivas. Através da ferramenta de customização disponível no site oficial[3] é possível selecionar quais componentes CSS e funcionalidades JavaScript irão fazer parte da aplicação.

Apresenta-se a estrutura de arquivos do Bootstrap na Figura 16.

[3] *http://getbootstrap.com/customize/*

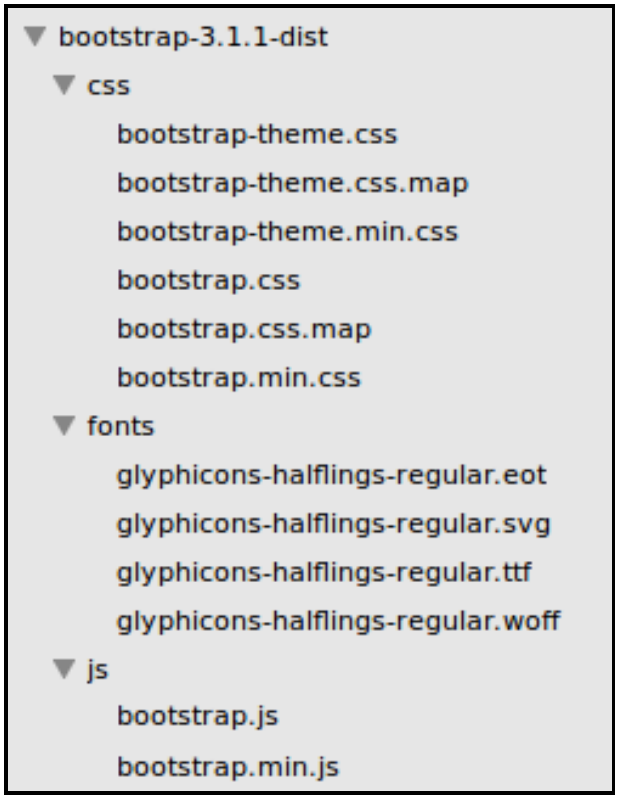

Figura 16 – Estrutura de Diretórios do Bootstrap

O Bootstrap é compatível com as últimas versões de diversos navegadores, tais como Internet Explorer, Mozilla Firefox, Google Chrome e Opera. Navegadores de dispositivos móveis também oferecem suporte ao Bootstrap. A partir da versão 2.0, o suporte ao *design* responsivo das páginas web foi integrado ao framework.

Para utilizar o Bootstrap, basta baixar o código fonte e adicioná-lo na página HTML. Outra opção seria referenciar uma CDN[4] confiável na página HTML. Esta segunda opção é viável quando não é necessário customizar os arquivos fontes. Ao término da instalação, o desenvolvedor adquire uma lista de componentes web, tais como, botões, tabelas, menus prontos para serem utilizados.

Na Figura 17, apresenta-se um exemplo enfatizando o uso do Bootstrap. Neste exemplo utiliza-se alguns componentes interessantes tais como, *navbar*, *jumbotron* e o sistema de *grid*.

[4] Content Delivery Network ou Rede de Fornecimento de Conteúdo

Figura 17 – Aplicação do Twiiter Bootstrap

Adaptação de *http* : *//bootstrapdocs.com/v*3.1.1*/docs/examples/jumbotron/*

Com a utilização do Twitter Bootstrap é possível desenvolver interfaces gráficas responsivas com qualidade e produtividade. O framework é atualizado com freqüência e possui uma documentação completa. Como o projeto é *open source*, possui ótimo suporte da comunidade.

3.3 Motivações

Os motivos que nortearam a escolha dos padrões, metodologias e tecnologias analisados anteriormente são apresentados na Tabela 10.

Tabela 10 – Decisões Tomadas.

Padrões, Metodologias e Tecnologias	**Motivo**
Scrum	Devido a complexidade e tamanho da ferramenta, é necessário uma abordagem que permita a elaboração de ciclos de desenvolvimento curtos. A partir de cada ciclo, é necessário que o produto resultante seja revisado. Ao término de cada revisão, mudanças no escopo do projeto poderão ocorrer.
Orientação a Meta	A orientação a metas facilita na identificação das necessidades omitidas em uma especificação de requisitos. Conhecendo as reais necessidades dos requisitos é possível definir melhores soluções que podem ser avaliadas a partir de critérios de qualidade. Ao longo da elicitação, diversas validações são realizadas. Assim, requisitos inconsistentes expressadas pelos atores são identificados e avaliados rapidamente.
Grails	O framework web Grails oferece um ambiente completo de desenvolvimento. Através da funcionalidade de *scaffolding*, é possível ofertar as funcionalidades de CRUD de uma entidade de domínio automaticamente. Este framework será utilizado neste trabalho para a construção da API REST de persistência da aplicação.

Continua na Próxima Página

Tabela 10 – *Continuação da Página Anterior*

Padrões, Metodologias e Tecnologias	**Motivo**
Twitter Bootstrap	O Twitter Bootstrap oferece um conjunto de recursos gráficos que irá facilitar no desenvolvimento das interfaces gráficas de uma aplicação. A principal finalidade do Twitter Bootstrap é consumir o menor tempo possível no desenvolvimento de uma aplicação web, seja ela uma página simples estática ou um grande portal dinâmico.
Node.js	O Node.js é uma plataforma de desenvolvimento destinada a aplicações em tempo real. A aplicação a ser desenvolvida possuirá um grande fluxo de dados entre o cliente e o servidor e deve exibir as informações aos usuários de forma assíncrona. Estas premissas são tratadas com excelência pelo Node.js.

4 Processo Metodológico

Neste capítulo, apresenta-se o processo metodológico elaborado para concretizar os objetivos definidos neste trabalho. Para isto, um processo de desenvolvimento para a ferramenta de argumentação foi definido.

4.1 Descrição do Processo

Para o êxito deste trabalho, um conjunto de tarefas fundamentais foram definidas. Apresenta-se na Tabela 11 uma descrição sucinta de cada tarefa.

Tabela 11 – Tarefas para o Êxito do Trabalho.

ID	**Tarefa**	**Descrição**
T1.	Analisar os conceitos de uma argumentação.	Esta tarefa visa definir conceitualmente e entender os principais elementos de uma argumentação. Nesta tarefa serão identificadas as lacunas e diretrizes úteis para o desenvolvimento da ferramenta.
T2.	Analisar modelos de argumentação.	Como a argumentação é um ramo interdisciplinar, diversas abordagens foram propostas. O objetivo desta tarefa é averiguar abordagens importantes para a história da argumentação, bem como novas abordagens com propostas promissoras.
T3.	Pesquisar ferramentas de argumentação.	Esta tarefa visa avaliar a consistência de cada ferramenta de argumentação. Para isto, uma lista de critérios foi elaborada. Ao término da avaliação, um conjunto de ideias bem como possíveis erros a serem evitados serão identificados para auxiliar o bom andamento deste trabalho.

Continua na Próxima Página

Tabela 11 – *Continuação da Página Anterior*

ID	Tarefa	Descrição
T4.	Elicitar requisitos.	Através da base de conhecimento obtida nas tarefas anteriores, possíveis requisitos serão elicitados. Para isto, técnicas orientadas à meta serão utilizadas visando explorar as características sociais identificadas em uma argumentação.
T5.	Definir Arquitetura de Software.	Com base nos requisitos funcionais e não funcionais elicitados, uma arquitetura de software será projetada. As principais decisões de projeto e os padrões arquiteturais serão escolhidos visando atender de forma eficiente todas as necessidades levantadas.
T6.	Implementar solução.	Desenvolver os requisitos elicitados com base na arquitetura de software definida.
T7.	Realizar testes na ferramenta.	Testar a nível unitário o código da ferramenta desenvolvida. Esta tarefa é essencial para a realização de futuras refatorações no código.
T8.	Implantar e avaliar ferramenta.	Ao término do desenvolvimento, a ferramenta será implantada e disponibilizada para uso. As críticas e sugestões de melhorias serão documentadas e servirão como insumo para futuras manutenções.

4.2 Metodologia de Desenvolvimento

Para o desenvolvimento da ferramenta de argumentação, um processo baseado em metodologias ágeis foi elaborado. O framework Scrum 3.1.1 foi selecionado para guiar a execução deste trabalho. Levando em conta um prazo de execução curto e a complexidade do produto, é necessário uma abordagem flexível e adaptativa, permitindo um desenvolvimento iterativo e incremental. Apresenta-se na Figura 18 o processo que foi adotado neste trabalho.

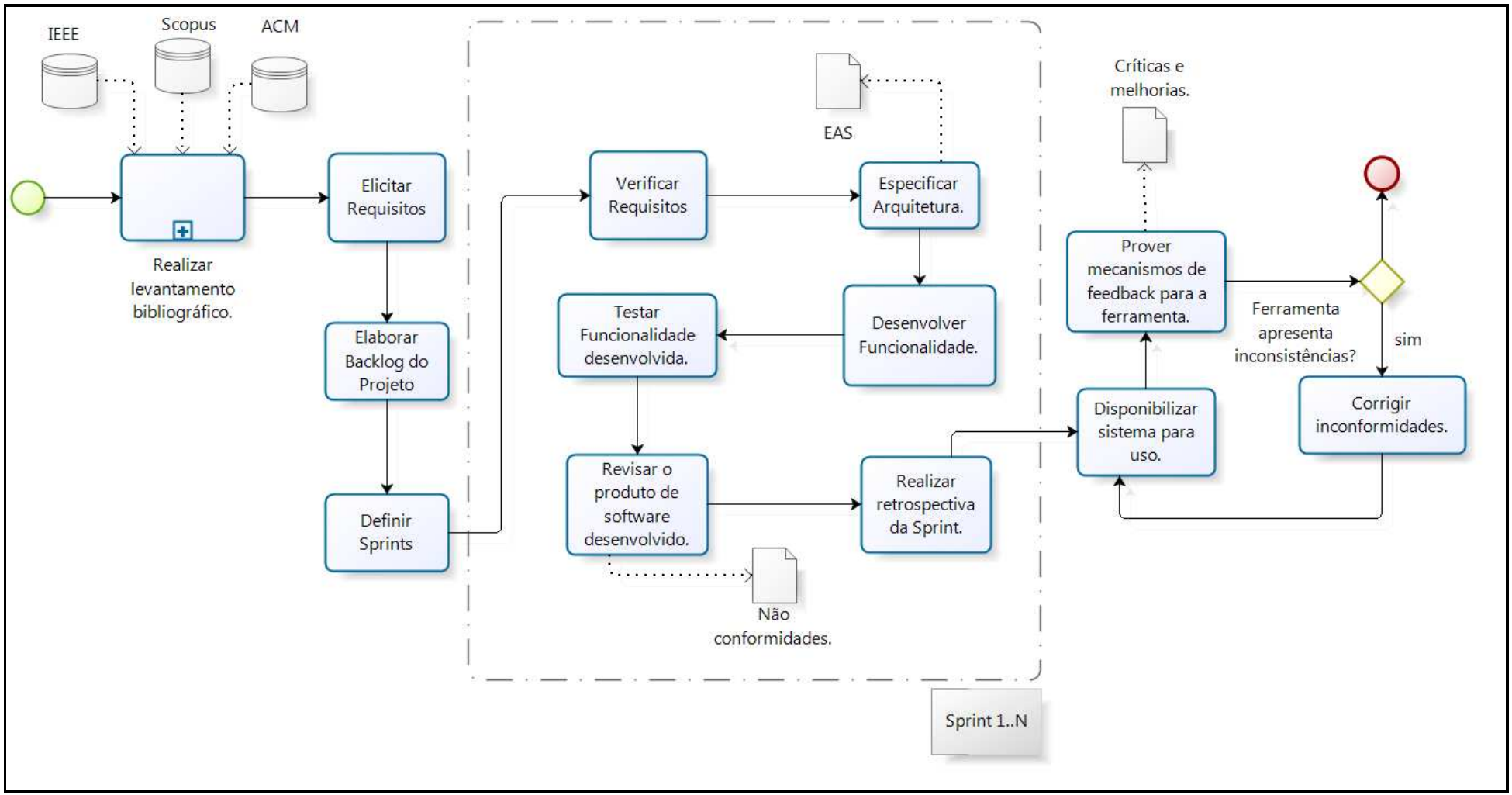

Figura 18 – Processo de Desenvolvimento.

As tarefas fundamentais descritas na Tabela 11 serviram de base para o processo de desenvolvimento apresentado na Figura 18. O processo inicia com uma contextualização dos aspectos gerais e específicos de uma argumentação. Para isto, realiza-se um levantamento bibliográfico em bibliotecas digitais consolidadas. Mais detalhes desta tarefa são apresentados na Figura 19, que exibe o subprocesso definido. Com a base de conhecimento construída, os requisitos de software foram elicitados. A partir dos requisitos, o *backlog* do produto foi formalizado e dividido em *Sprints*. Cada *Sprint* possui atividades para a verificação dos requisitos, especificação de arquitetura, implementação e a realização de testes. Ao término das *Sprints* a ferramenta desenvolvida foi disponibilizada para uso. Mecanismos de *feedback* para identificar críticas e obter sugestões de melhoria foram planejados.

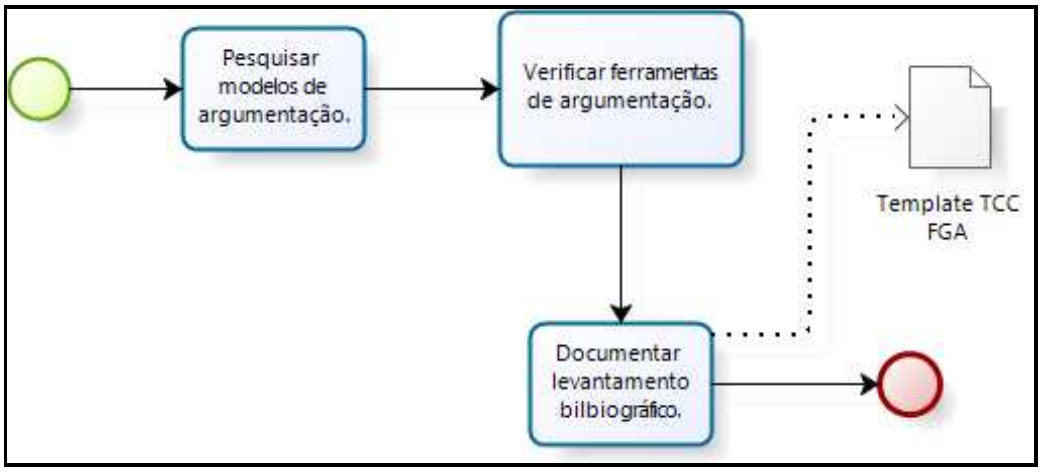

Figura 19 – Subprocesso: Realizar levantamento bibliográfico

Apresenta-se na Tabela 12, as entradas, saídas e tecnologias utilizadas para a execução das atividades exibidas na Figura 18.

Tabela 12 – E/S e Tecnologias Utilizadas

Atividades	**Entrada**	**Saída**	**Tecnologias**
Realizar Levantamento Bibliográfico	Bibliotecas Digitais (ACM, Scopus e IEEE)	Revisão Bibliográfica e Avaliação das Ferramentas de Argumentação	Zootero

Continua na Próxima Página

Tabela 12 – *Continuação da Página Anterior*

Atividades	**Entrada**	**Saída**	**Tecnologias**
Elicitar Requisitos	Revisão Bibliográfica e Avaliação das Ferramentas de Argumentação	Modelo de Raciocínio Estratégico (SR) e Especificação Suplementar	Framework I* 3.1.2 e OpenOME[1]
Elaborar *Backlog* do Projeto	Modelo (SR) e Especificação Suplementar	*Backlog* do Produto	Trello[2]
Definir *Sprints*	*Backlog* do Produto	*Sprints* Planejadas	Trello
Verificar Requisitos	*Backlog* da *Sprint*	Priorização dos Requisitos	Trello
Especificar Arquitetura	*Backlog* da *Sprint*	Especificação da Arquitetura de Software (EAS)	Microsoft Visio
Desenvolver Funcionalidade	*Backlog* da *Sprint* e EAS	Código Implementado	Grails 3.2.1, Node.js 3.2.2 e Sublime Text[3]
Testar Produto de Software Desenvolvido	Código Desenvolvido	Testes Unitários	JUnit, Spock e Node-unit
Revisar Produto de Software Desenvolvido	Funcionalidades Concluídas	Críticas e Melhorias	*Checklists*
Realizar Retrospectiva da *Sprint*	*Sprint* Concluída	Lições para o planejamento de novas *Sprints*	SenseiTool[4]
Prover Mecanismos de *Feedback* para a Ferramenta	-	Questionários e Grupos de Suporte	Google Docs e Gmail

Continua na Próxima Página

[1] Ferramenta para a modelagem social. Disponível em: *https* : *//se.cs.toronto.edu/trac/ome/wiki*
[2] Ferramenta para gerenciamento de projetos. Disponível em: *https* : *//trello.com/*
[3] Editor de Texto. Disponível em: *http* : *//www.sublimetext.com/*
[4] Ferramenta para a realização de retrospectivas. Disponível em: *http* : *//beta.senseitool.com/home*

Tabela 12 – *Continuação da Página Anterior*

Atividades	**Entrada**	**Saída**	**Tecnologias**
Disponibilizar Sistema para Uso	Aplicação Concluída	*Deployment* da Aplicação	Apache Tomcat 7 e OpenShift[5]
Corrigir Inconformidades	Sugestões de Melhorias e Críticas	Planejamento de novas Sprints	-

4.3 Etapa 01 - Referencial Teórico

O referencial teórico consiste de uma revisão dos trabalhos existentes a partir de um tema abordado. Esta revisão pode ser feita em livros, artigos, monografias, teses e outras bases científicas confiáveis. Segundo LAKATOS e MARCONI (2010), o referencial teórico permite verificar o estado atual do problema a ser pesquisado.

As bases de pesquisa utilizadas neste trabalho, foram as seguintes bibliotecas digitais:

- ACM Digital Library: <http://dl.acm.org/>;
- Scopus: <http://www.scopus.com/>;
- IEEE Xplore: <http://ieeexplore.ieee.org/Xplore/home.jsp>;
- Google Scholar: <http://scholar.google.com.br/>.

Os objetos de pesquisa foram os modelos e frameworks de argumentação que têm como objetivo formalizar a comunicação entre as pessoas. Os modelos e frameworks foram estudados e uma análise de viabilidade foi realizada 2. Como resultado, as abordagens encontradas serviram como insumo para a engenharia de requisitos da ferramenta de argumentação que foi desenvolvida.

4.4 Etapa 02 - Especificação da Solução

Com base no referencial teórico levantado, o framework de argumentação ACE mostrou-se adequado para servir como base de conhecimento no desenvolvimento da ferramenta. O ACE oferece uma linguagem baseada em grafos e disponibiliza algoritmos utilizados para avaliar a aceitabilidade das informações presentes em uma argumentação.

[5] Serviço de *Hosting* para Node.js. Disponível em: $https://www.openshift.com/developers/node-js$

Os requisitos foram elicitados utilizando a engenharia de requisitos orientadas à meta e histórias de usuário. A orientação à meta foi utilizada para os requisitos mais complexos relacionados a execução da argumentação na ferramenta. O modelo resultante foi mapeado para histórias de usuários em conjunto com outros requisitos de baixa complexidade.

Como diversos modelos de argumentação foram analisados no referencial teórico, algumas questões de pesquisa tinham mais de uma solução. Utilizando o framework I Star 3.1.2, foi possível definir critérios de qualidade que auxiliaram na escolha da solução mais viável para uma determinada meta. A partir do framework I Star, as principais funcionalidades da ferramenta a ser desenvolvida foram identificadas e avaliadas. A elicitação de requisitos seguiu as seguintes etapas:

1. Identificação dos atores relacionados com a ferramenta: Nesta etapa foram identificados dois atores. O "usuário" que irá utilizar a ferramenta e o "sistema de argumentação" que irá suprir as funcionalidades da ferramenta.

2. Identificação das intenções dos atores: Nesta etapa, foram utilizados as análises realizadas no Capítulo 2.2. Cada modelo ou framework avaliado trata a argumentação a partir de um ponto de vista diferente. Porém, objetivos semelhantes foram identificados. Com base nestes objetivos, metas rígidas e flexíveis foram propostas para os atores "usuário" e "sistema de argumentação".

3. Definir relações de dependência entre os atores identificados: Nesta etapa, foram definidos os relacionamentos entre os atores. Essencialmente, foi especificado como o sistema de argumentação poderia oferecer suporte ao ator "usuário" na prática de discussões na internet.

4. Detalhar modelo: Nesta etapa, a rede de relacionamentos entre os atores foi especificada. Para cada meta definida, várias tarefas foram propostas para sua resolução. Isto permite a implementação de várias soluções diferentes para o mesmo problema. Para determinar quais soluções adotar, foram utilizadas as metas flexíveis. A partir dos critérios de qualidades definidos é possível identificar qual solução mais se adequa ao ambiente.

Como término das atividades acima, um modelo de dependência estratégica (SD) dos atores identificados foi elaborado e pode ser visto na seção **??**. Com base neste modelo, os requisitos elicitados foram convertidos para histórias de usuário. Para facilitar a interação entre o usuário e a ferramenta, outros requisitos não mapeados no modelo SD foram levantados através de histórias de usuário. Estes requisitos representam algumas funcionalidades herdadas de uma rede social como:

- Criar um perfil social: Este perfil é público e armazena as informações do usuário;
- Construir um vínculo de amizade: Um usuário pode fazer amigos e convidá-los para as suas argumentações;
- Criar grupos de argumentação: Grupo de usuários com interesses comuns;
- Criar categorias de argumentação: Isto facilita a organização das argumentações que estão sendo criadas.

4.4.1 Histórias de Usuário

Com base no Modelo 21, um conjunto de histórias foram definidas visando atender as metas elicitadas. Apresenta-se na Tabela 13, o *Backlog* do produto resultante desta análise.

Tabela 13 – Histórias de Usuário.

Identificador	**História de Usuário**
HI1	Eu, como usuário, desejo me cadastrar na ferramenta, para ter acesso as funcionalidades.
HI2	Eu, como usuário, desejo adicionar amigos, para que possa interagir com eles.
HI3	Eu, como usuário, gostaria de manter argumentações, para debater com meus amigos.
HI4	Eu, como usuário, desejo adicionar amigos na minha argumentação, para que eles possam participar.
HI5	Eu, como usuário, desejo manter grupos, para permitir a interação entre pessoas que não possuem vínculo de amizade.
HI6	Eu, como usuário, desejo visualizar a discussão em tempo real através de grafos, para que eu tenha conhecimento das informações presentes.
HI7	Eu, como usuário, desejo opinar nas argumentações, assim todos os outros participantes saberão o que eu penso.

Continua na Próxima Página

Tabela 13 – *Continuação da Página Anterior*

Identificador	**História de Usuário**
HI8	Eu, como usuário, desejo manter categorias, para classificar as argumentações que serão criadas.
HI9	Eu, como usuário, desejo listar todos os meus amigos, para ter consciência das pessoas que eu adicionei.
HI10	Eu, como usuário, desejo pesquisar argumentações, grupos e pessoas, para verificar a existência dos mesmos.
HI11	Eu, como usuário, desejo listar todas as minhas argumentações.
HI12	Eu, como usuário, desejo visualizar uma linha do tempo das opiniões de uma discussão, para saber quais foram os últimos dados inseridos.
HI13	Eu, como usuário, desejo exportar uma discussão para a ontologia AIF, assim poderei integrar minha discussão com outras ferramentas que são compatíveis com esta tecnologia.
HI14	Eu, como usuário, desejo listar todos os meus grupos, para ter consciência dos grupos que eu já adicionei.
HI15	Eu, como usuário, desejo visualizar a validade da minha opinião, para ter consciência do impacto da minha opinião na discussão.

Com base nos requisitos elicitados, as *Sprints* de desenvolvimento foram planejadas. Além disto, uma arquitetura de software foi elaborada para suprir os requisitos funcionais e não funcionais identificados. Esta arquitetura sofreu modificações na medida em que as sprints eram implementadas. A arquitetura final pode ser vista na seção 22.

4.5 Etapa 03 - Desenvolvimento da Solução

Com base na Tabela 13, foram definidas as seguintes *Sprints* a serem desenvolvidas no decorrer deste trabalho. Apresenta-se na Tabela 14 o escopo de cada *Sprint*.

Tabela 14 – Definição das *Sprints*.

Sprint	**História de Usuário**	**Início**	**Fim**
1	HI1 HI2	03/06/2014	18/06/2014
2	HI5 HI8	19/06/2014	04/07/2014
3	HI14 HI11	19/07/2014	30/07/2014
4	HI10 HI9	31/07/2014	14/08/2014
5	HI3 HI4	15/08/2014	25/08/2014
6	HI6 HI7 HI12	26/08/2014	15/09/2014
7	H15	15/09/2014	01/10/2014
8	HI13	01/10/2014	15/10/2014

As *sprints* um e dois tem como objetivo a implementação das funcionalidades relacionadas ao cadastro de usuários, grupos e categorias. A partir destas entidades os usuários podem interagir entre si através de vínculos de amizade e/ou grupos. As categorias são utilizadas para classificar as argumentações criadas pelo usuário. As *sprints* três e quatro tem como objetivo o desenvolvimento das funcionalidades de exibição dos dados presentes na ferramenta. As *sprints* cinco e seis focam no desenvolvimento do processo de argumentação. O usuário pode criar argumentações, convidar participantes e interagir cmom os mesmos em tempo real de forma gráfica.

A argumentação foi implementada através de grafos de acordo com o framework ACE 2.2.4. A biblioteca Vis.js foi utilizada para exibir os grafos e uma linha do tempo para enfatizar o período no qual os argumentos foram adicionados.

O comportamento em tempo real dos grafos de argumentação foi implementado da seguinte forma:

1. O protocolo *web socket* foi utilizado através da biblioteca *socket.io* para realizar a comunicação entre o cliente e o servidor de forma assíncrona.

2. Após a autenticação, o usuário irá se conectar no servidor *web socket*. A sala *room* padrão é chamada *lobby*;

3. Para cada argumentação é criada uma sala *room* no servidor *web socket*;

4. Os usuários serão adicionados nas salas de acordo com a argumentação que os mesmos estão participando;

5. Os participantes da argumentação interceptam eventos assíncronos contendo os novos argumentos adicionados;

6. Com base nos eventos acima, o grafo de argumentação é atualizado. Além disto, uma linha do tempo foi adicionada para exibir os argumentos dos usuários temporalmente.

A *sprint* sete tem como objetivo a implementação dos algoritmos utilizados para avaliar a aceitabilidade dos argumentos. Para verificar a aceitabilidade de um argumentos os seguintes passos são necessários:

1. Considerando um argumento a ser avaliado p, deve-se identificar todos os argumentos que se relacionem com p. Para isto deve-se executar uma busca para obter todos os sub grafos que terminem em p. Como o grafo é direcionado, é importante conhecer apenas os nós que conseguem chegar em p;
2. Com o grafo de argumentação baseado em p definido, deve-se identificar os componentes fortemente conectados que compõe o grafo. Para isto, pode-se utilizar o Algoritmo de Tarjan;
3. Realizar a ordenação topológica entre os componentes fortemente conectados. Assim, a ordem de avaliação dos componentes é identificada;
4. Avaliar cada componente de acordo com a ordenação topológica. Se o componente foi acíclico, o mesmo terá apenas um argumento e deve ser avaliado de acordo com a política de rotulação do framework ACE. Já os componentes cíclicos são avaliados da seguinte forma:
   a) Identificar todos os ciclos elementares do componente cíclico;
   b) Rotular como aceito todos os argumentos do componente cíclico;
   c) Percorrer cada ciclo elementar e rotular de acordo com as políticas do framework ACE.
5. Após finalizar a avaliação, o canal de comunicação assíncrono é utilizado para enviar os resultados deste procedimento aos participantes da argumentação.

A *sprint* 8 tem como objetivo a interoperabilidade da ferramenta com outros sistemas de argumentação. A partir de serviços RESTful as informações persistidas podem ser compartilhadas com outras ferramentas.

4.6 Etapa 04 - Cenário de Uso

Para validar a ferramenta foi realizado um experimento baseado na técnica de pesquisa-ação. A técnica de pesquisa-ação é um processo sistemático, contínuo e empírico que tem como objetivo o aprimoramento da prática (TRIPP, 2005).

4.6.1 Grupo

O grupo de usuários que testou a ferramenta é composto por oito estudantes do curso de Engenharia de Software do sexto, nono e décimo semestre. Apresenta-se na Tabela 15, os subgrupos categorizados com base no semestre corrente de cada participante do experimento.

Tabela 15 – Grupos de Usuários.

Grupo	**Semestre**	**Participantes**
01	6º	2
02	9º	3
03	10º	3

4.6.2 Etapas do Experimento

A pesquisa-ação é composta de uma espiral retroalimentada contendo quatro etapas. Apresenta-se na Figura 20 as etapas desta técnica.

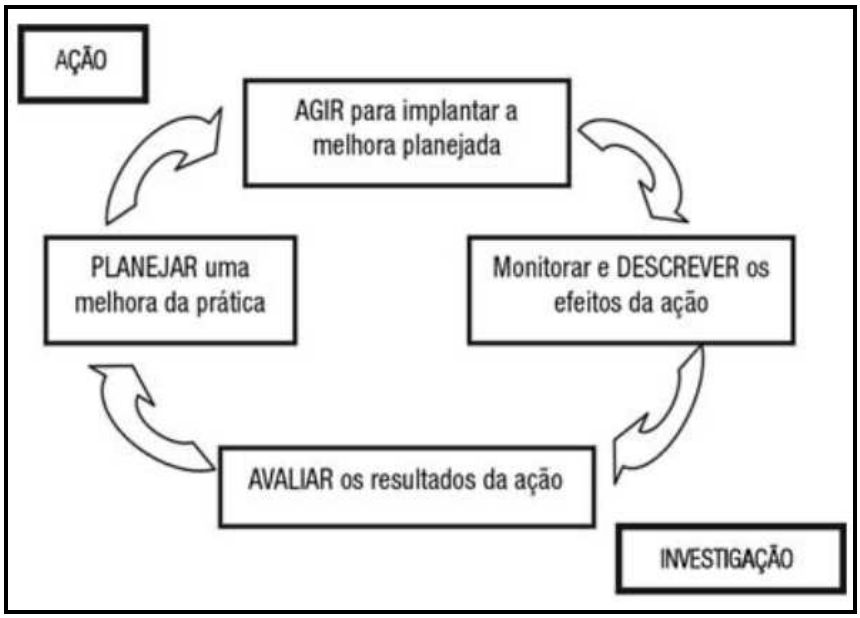

Figura 20 – Etapas da Pesquisa-ação

Fonte: (TRIPP, 2005).

4.6.3 Condução

A fase de planejamento tem como objetivo definir uma melhoria prática em determinado contexto. A ferramenta ACE-CAST representa esta possível melhoria nas argumentações onlines realizadas por diversos usuários. Após o uso da ferramenta, o relato

de cada usuário foi avaliado e monitorado através de uma ficha de avaliação contendo as perguntas descritas na Tabela 16.

Tabela 16 – Questionário.

Identificador	**Pergunta**	**Resposta**
01	Costuma participar de argumentações online?	Sim ou Não
02	Criar uma argumentação?	1 - Muito Difícil, 2 - Difícil, 3 - Moderado, 4 - Fácil, 5 -Muito Fácil
03	Adicionar participantes na argumentação?	1 - Muito Difícil, 2 - Difícil, 3 - Moderado, 4 - Fácil, 5 -Muito Fácil
04	Adicionar argumentos na discussão desejada?	1 - Muito Difícil, 2 - Difícil, 3 - Moderado, 4 - Fácil, 5 -Muito Fácil
05	Verificar a validade dos argumentos na discussão desejada?	1 - Muito Difícil, 2 - Difícil, 3 - Moderado, 4 - Fácil, 5 -Muito Fácil
06	Customizar a interface gráfica da ferramenta?	1 - Muito Difícil, 2 - Difícil, 3 - Moderado, 4 - Fácil, 5 -Muito Fácil
06	Impressões Finais?	Discursiva.

As perguntas descritas na Tabela 16 possuem ênfase qualitativa. O objetivo é identificar as impressões iniciais dos usuários que utilizaram a ferramenta.

5 Resultados

A partir das tarefas definidas no Capítulo 4, o projeto foi finalizado. Neste Capítulo, apresentam-se os principais resultados obtidos. Na seção 5.1, apresentam-se os requisitos elicitados que conduzem o desenvolvimento da ferramenta. Na seção 22, é definida uma arquitetura de software capaz de suportar as necessidades levantadas. Na seção 5.3, apresentam-se as funcionalidades da ferramenta ACE-CAST. Na seção 4.6, apresentam-se os resultados obtidos na validação da ferramenta.

5.1 Requisitos do Sistema

A modelagem de requisitos foi realizada utilizando técnicas orientadas à meta 3.1.2 e histórias de usuário. O objetivo das técnicas orientadas à metas foi a elicitação dos requisitos mais complexos. Ao término da elicitação o modelo resultante será convertido para histórias de usuário visando a criação de um *Backlog* do produto.

5.1.1 Elicitação Orientada à Meta

Para concretizar os resultados da elicitação de requisitos, apresenta-se na Figura 21 o modelo de raciocínio estratégico.

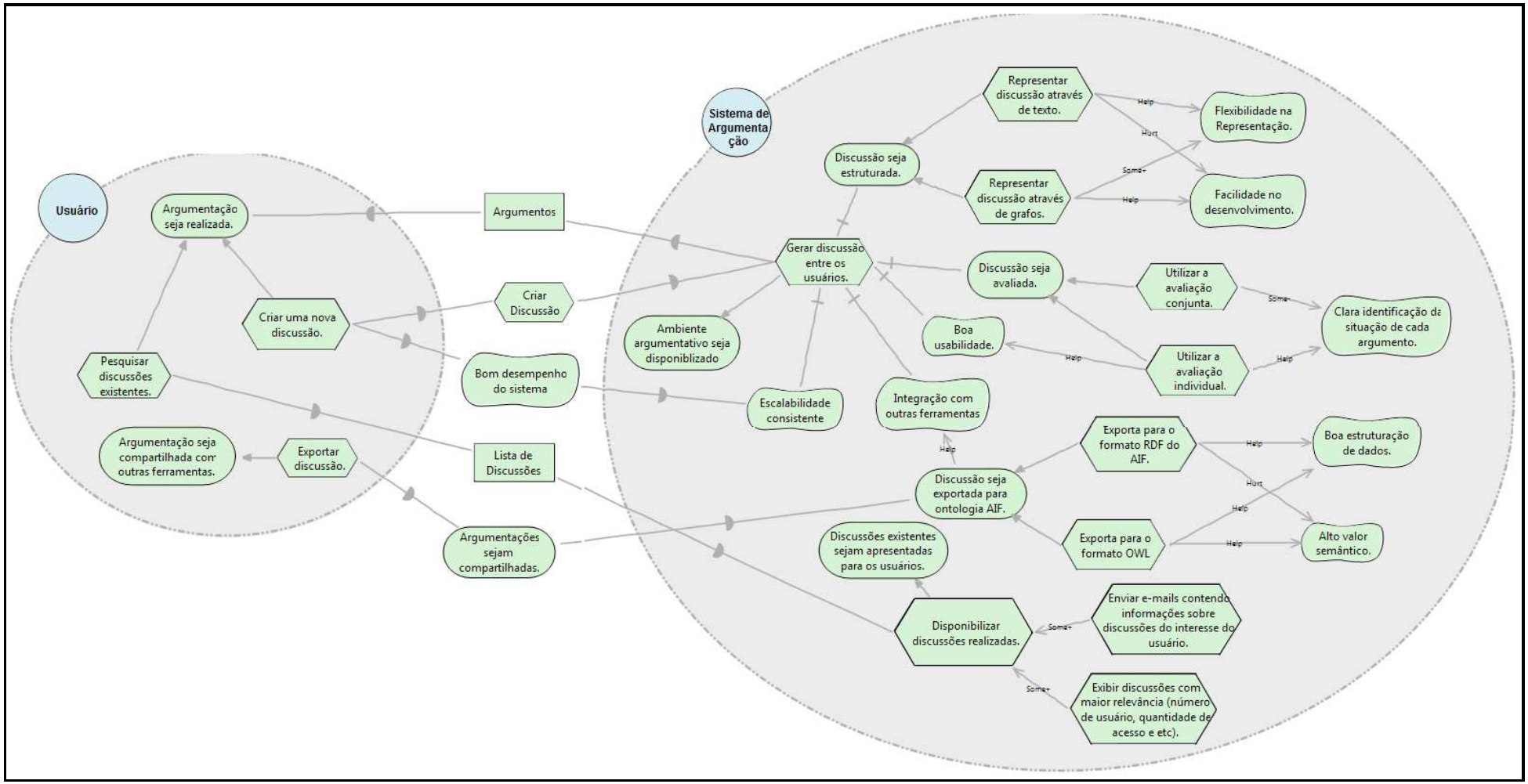

Figura 21 – Modelo de Raciocínio Estratégico.

Observando a Figura 21, é possível identificar soluções consistentes que atendem os critérios de qualidade definidos. Para começar, a argumentação será representada por meio de grafos. Os grafos facilitam a visualização da discussão, bem como no desenvolvimento de procedimentos lógicos para a análise da aceitabilidade dos argumentos. A aceitabilidade dos argumentos será determinada através da avaliação individual. Para usuários leigos, no contexto da argumentação, as abordagens conjuntas dificultam a análise do discurso. A argumentação será exportada para o formato OWL. O formato OWL permite a construção de ontologias contendo regras semânticas que podem auxiliar na avaliação estrutural e lógica das discussões.

Como a ferramenta de argumentação será colaborativa, é necessário que aspectos como escalabilidade e usabilidade sejam bem projetados. A solução para otimizar o desempenho do sistema será a plataforma Node.js 3.2.2. Enquanto que para a usabilidade do sistema, serão utilizados os componentes responsivos do Framework Twitter Bootstrap 3.2.3, bem como bibliotecas JQUERY para a apresentação dos dados.

5.2 Arquitetura de Software

Para suprir os requisitos elicitados, foi elaborada uma arquitetura de software seguindo o padrão arquitetural cliente-servidor. A arquitetura é baseada no padrão arquitetural MVC e é composta dos pacotes: *Core* e *Modules*. O módulo *Core* é desenvolvido a partir do plataforma Node.js 3.2.2. Os papeis deste módulo são descritos logo abaixo:

- API REST para as operações de CRUD das entidades definidas na camada de modelo. Assim, outras aplicações poderão se comunicar com a ferramenta de argumentação.

- Gerenciamento das requisições realizadas pelos usuários. Estas requisições podem ser HTTP ou seguir o protocolo de *Web Sockets*. O protocolo de *Web Sockets* será utilizado para realizar comunicações em tempo real entre o cliente e o servidor;

- As regras de negócio da ferramenta serão implementadas neste módulo.

O pacote *Modules* é um aglomerado de módulos externos que serão utilizadas no módulo *Core*. Estes módulos são instalados através do gerenciador de pacotes NPM[1] e permitem estender diversas funcionalidades não atendidas de forma padrão pelo Node.js. Os principais módulos utilizados são:

[1] https://www.npmjs.org/

- Express.js: Framework de desenvolvimento web para o Node.js. Este framework de desenvolvimento provê recursos para a construção de aplicações web seguindo o padrão arquitetural MVC;
- Socket.io: Possibilita comunicação bidirecional orientada a eventos em tempo real. Este módulo será útil na implementação de funcionalidades dinâmicas;
- Passport.js: Framework para garantir a segurança de acesso da aplicação;
- Mongoose.js: Ferramenta que permite a modelagem de objetos baseados no banco de dados Mongodb.

A lista completa de todos os módulos utilizados está disponível no arquivo *package.json* que se encontra na raiz do código fonte. Com este arquivo o gerenciado NPM consegue efetuar o download de todas as dependências do projeto.

O pacote *Core* contêm três camadas: *Views*, *Routes* e *Models*. A camada *Views* contêm todas as páginas de visualização que serão apresentadas na aplicação. A camada *Routes* realiza o gerenciamento das requisições realizadas pelo cliente, caso necessário ele acessa o módulo *lib* que contêm as regras de negócio da aplicação. Para finalizar, a camada *Models* contêm todas as regras de persistência utilizadas na aplicação. Apresenta-se na Figura 22 a arquitetura definida.

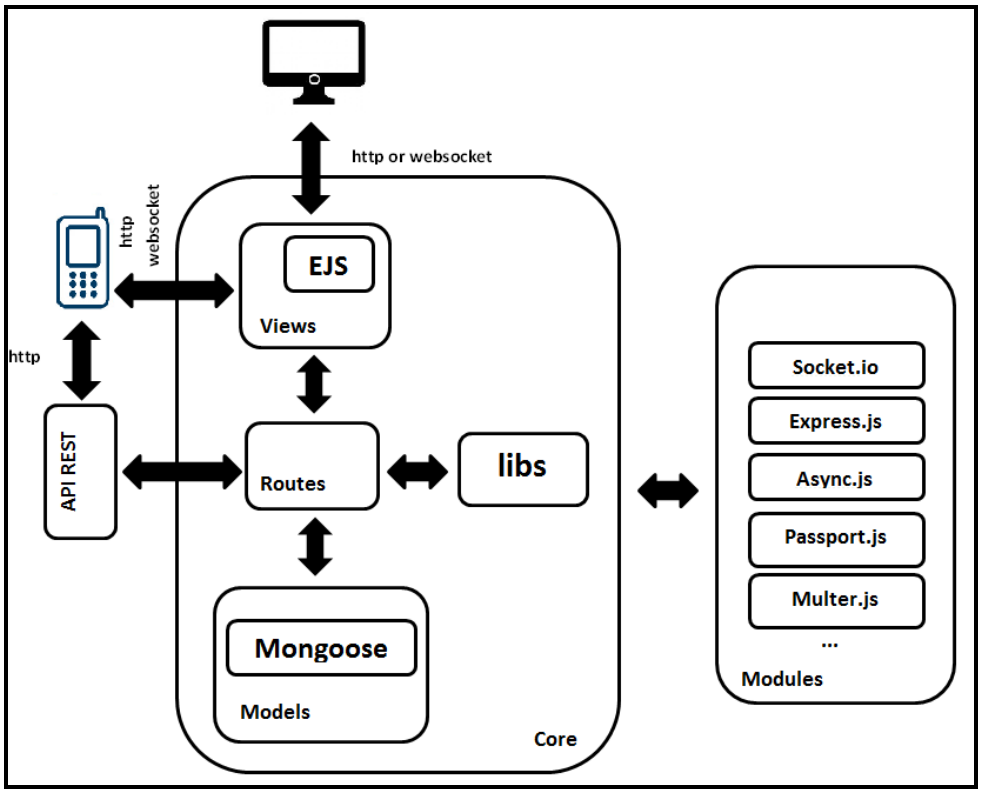

Figura 22 – Arquitetura do ACE-CAST

Para guiar o desenvolvimento da API, foi projetado um diagrama de classes baseado na linguagem UML, que enfatiza as entidades presentes na engenharia de domínio da ferramenta. O diagrama é apresentado na Figura 23.

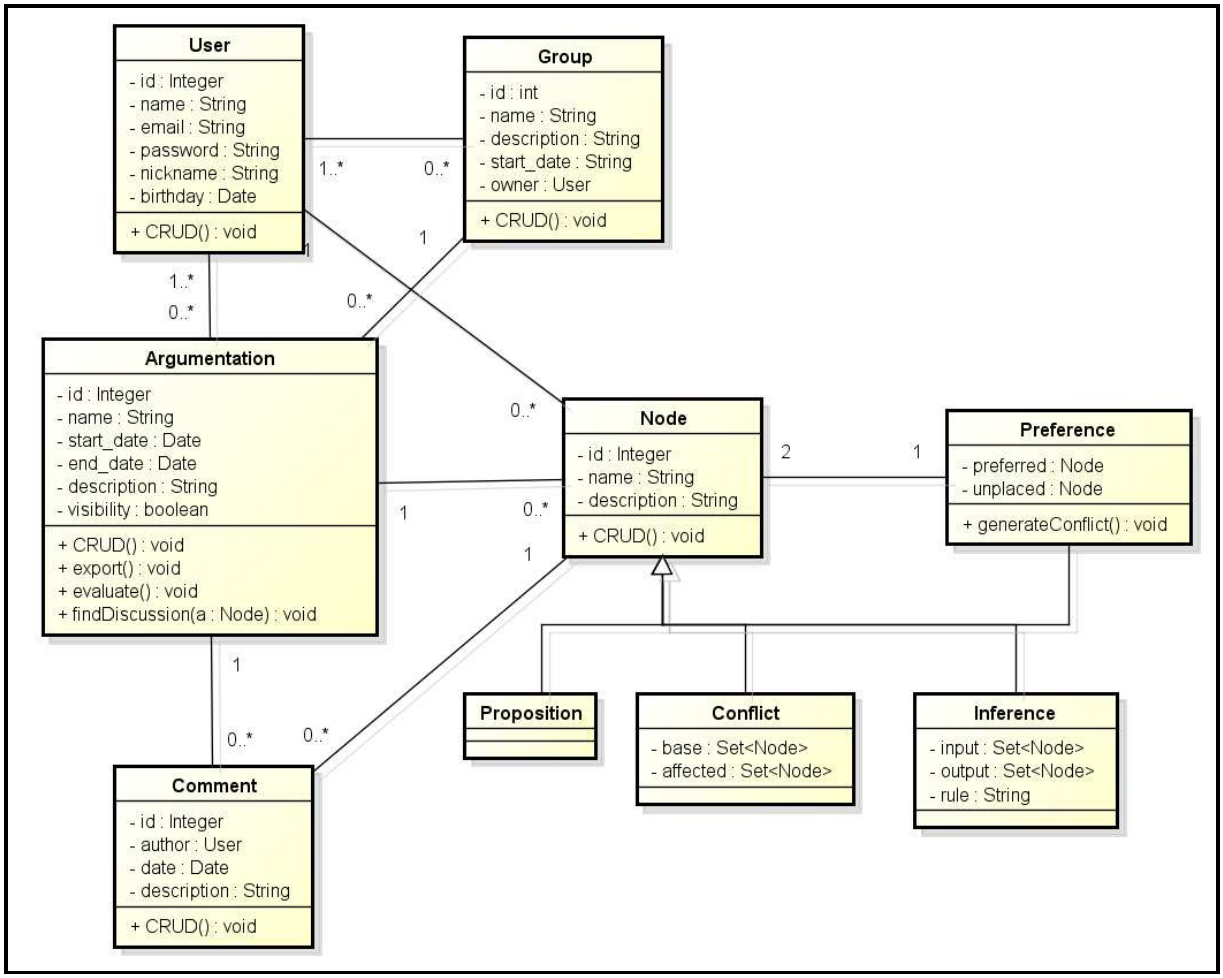

Figura 23 – Diagrama de Classes da API

Com base na Figura 23, o diagrama de classes foi construído baseado nas diretrizes definidas no Framework ACE 2.2.4. A classe argumentação representa um grafo direcionado. A partir deste grafo, vários nós (vértices do grafo), podem ser estabelecidos. Uma herança foi utilizada para representar os diferentes tipos de nó disponíveis em uma discussão. É interessante ressaltar que os nós foram caracterizados com base na ontologia AIF 2.3. Para permitir a colaboração entre os usuários da ferramenta, grupos de argumentação foram criados. A visibilidade da argumentação pode ser pública ou restrita a apenas um grupo. Para finalizar, é possível registrar comentários acerca das argumentações realizadas.

5.3 Ferramenta ACE-CAST

Para começar a utilizar a ferramenta é necessário que o usuário esteja cadastrado. Para realizar o cadastro na ferramenta, basta clicar no link *crie uma conta* exibido na Figura 24.

Figura 24 – Página de Login

Após solicitar o cadastro, o usuário será redirecionado para um formulário onde o mesmo deve informar seus dados pessoais. Após preencher o formulário, basta clicar no botão *Registrar* para finalizar seu cadastro. Após o registro, o usuário será redirecionado para a página inicial da ferramenta. A página inicial da ferramenta é apresentada na Figura 25.

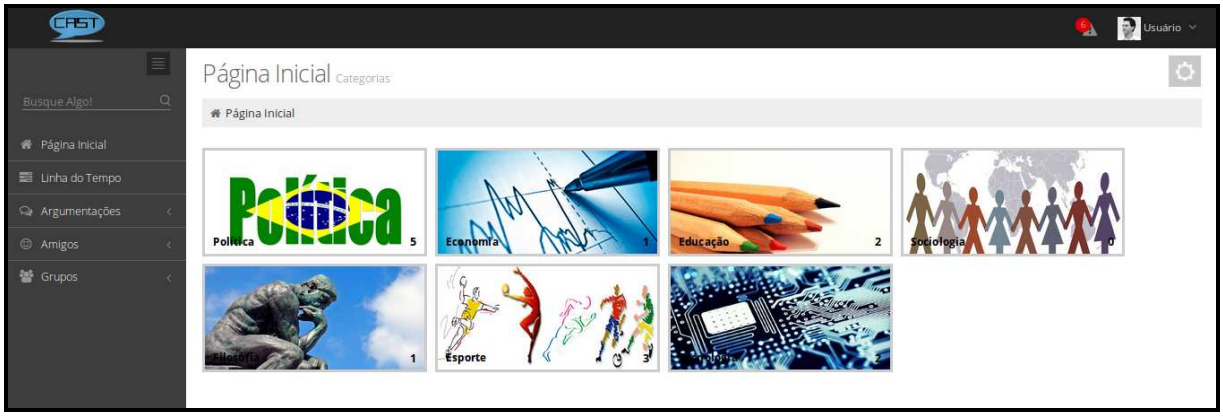

Figura 25 – Página Inicial

Na página inicial, o usuário possui acesso a todas as argumentações associadas a ele. Para facilitar a busca e a visualização, as argumentações foram agrupadas em categorias. Caso o usuário deseje, o mesmo poderá criar um nova categoria.

Para criar uma argumentação, é necessário preencher o formulário da Figura 26 contendo as informações básicas da argumentação.

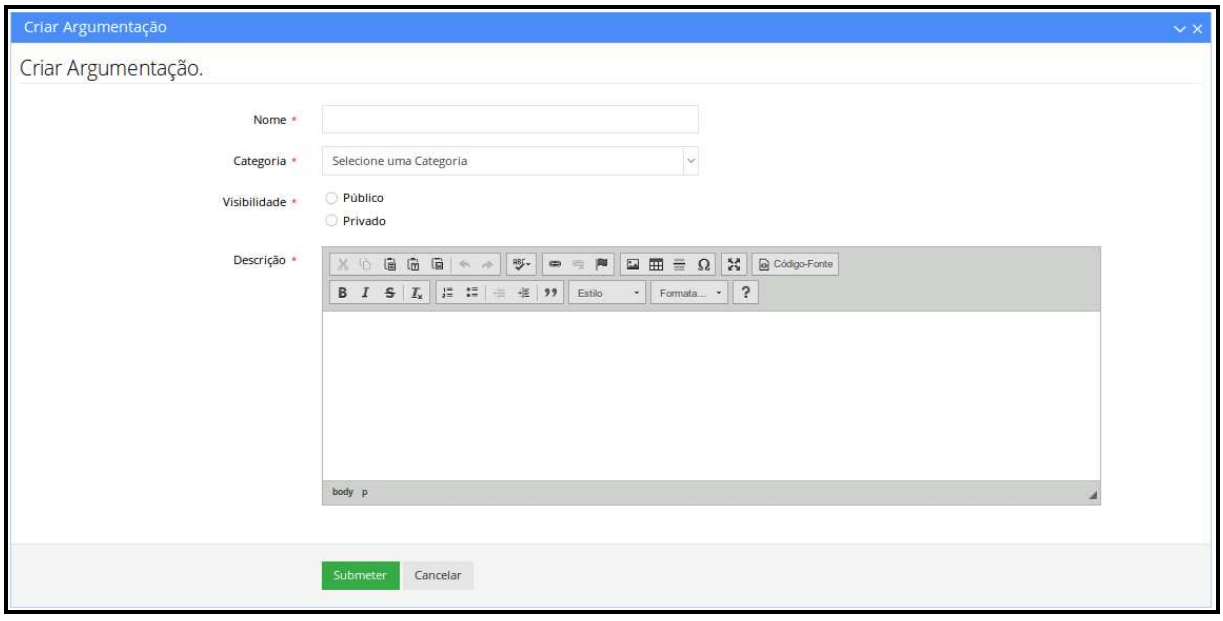

Figura 26 – Criar uma Argumentação

Após criar uma argumentação, o usuário é redirecionado para a página de visualização onde ele pode adicionar participantes a argumentação. Lembrando que caso a argumentação tenha sido criada por um grupo, todos os participantes do grupo já são

adicionados no momento da criação. A página de adição de participantes em uma argumentação é apresentada na Figura 27.

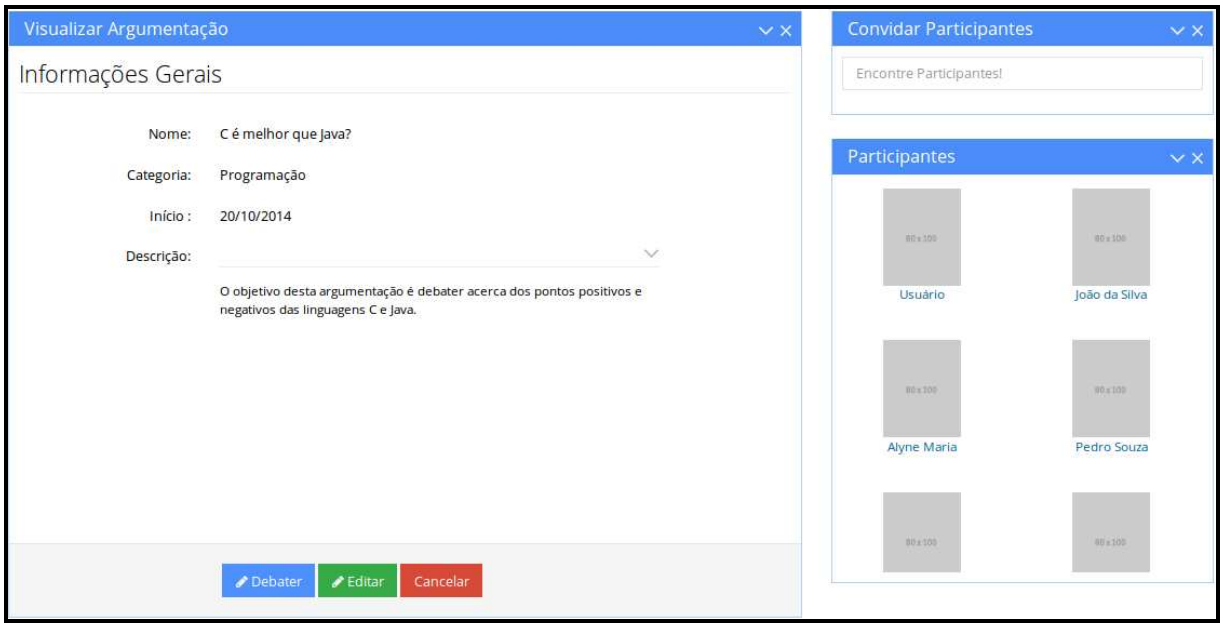

Figura 27 – Adicionar Usuários na Argumentação

Para o usuário inserir argumentos na argumentação criada, basta clicar no botão debater. Após o clique, o usuário será redirecionado para a página de debate. Nesta página, será apresentado ao usuário o grafo de argumentação em tempo real Figura 28. Além disto, uma linha do tempo auxiliará na visualização dos últimos argumentos adicionados Figura 29.

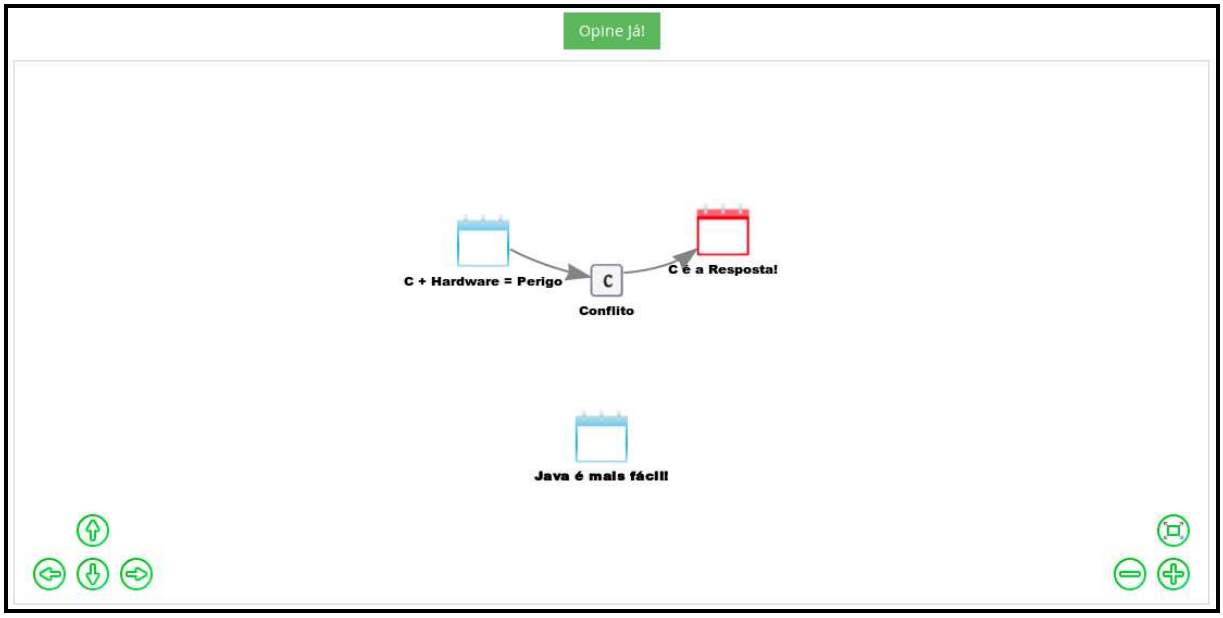

Figura 28 – Grafo da Argumentação em Tempo Real

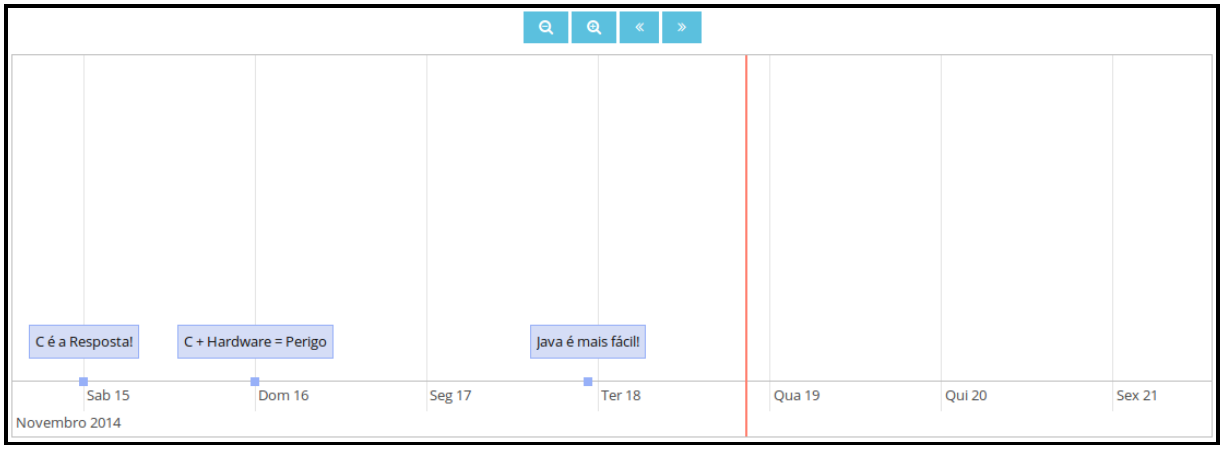

Figura 29 – Linha do Tempo da Argumentação em Tempo Real

Para criar uma opinião, o usuário deve clicar em *Opine Já* e um formulário do tipo *wizard* será aberto para que o usuário digite as informações de seu argumento. Para interagir com os argumentos já criados, o usuário deve posicionar o mouse acima do argumento desejado e escolher a opção desejada:

- O botão *veja mais* permite que o usuário visualize o argumento com mais detalhes;
- O botão verde deve ser utilizado para criar um novo argumento que tenha este como suporte;

- O botão vermelho deve ser utilizado para criar um novo argumento que tenha este como conflito.

Na Figura 30 é apresentada as opções disponíveis a partir dos argumentos já criados.

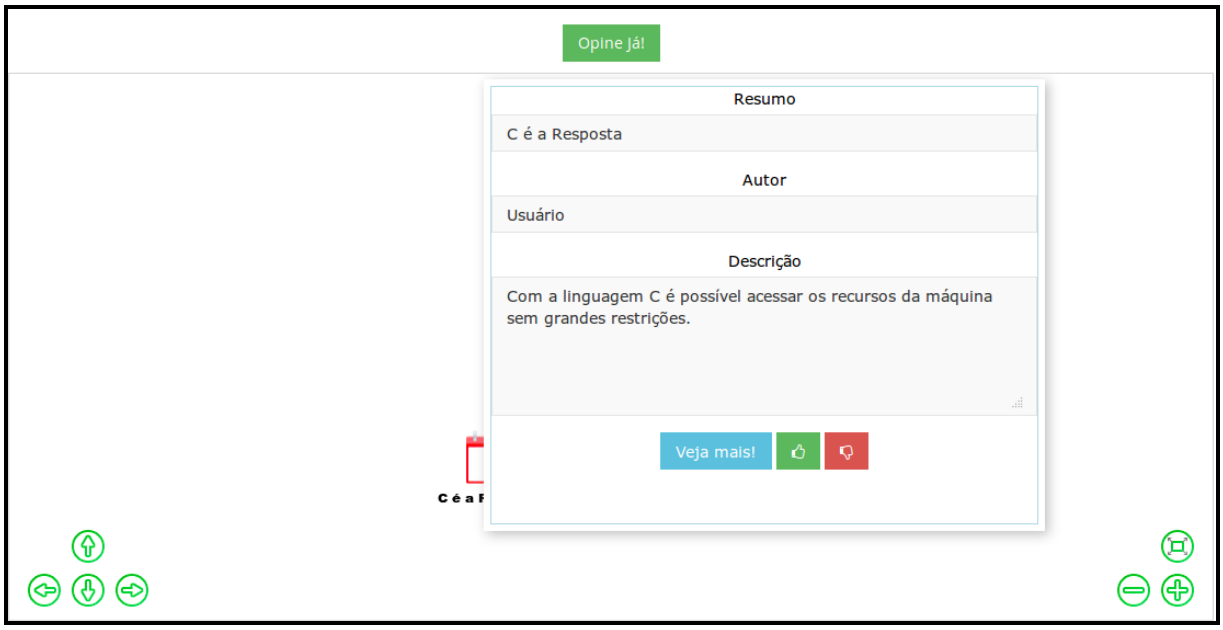

Figura 30 – Opções dos Argumentos Existentes

Os argumentos de opinião são representados através de *post its*. As cores representam a validade de determinado argumento. O *post it* de cor azul infere que o argumento está aceito, ou seja, válido. O *post it* de cor vermelha infere que o argumento esta sendo atacado, ou seja, invalido. E, para finalizar, o *post it* de cor amarela infere que o argumento foi atacado por uma preferência. Já os argumentos de inferência, são representados por um circulo contendo a sua letra inicial.

5.3.1 API RESTful

Os dados persistidos na aplicação poderão ser acessados através da API REST que foi desenvolvida. Apresenta-se na Figura 31, a execução do serviço que lista as argumentações presentes no banco de dados da ferramenta.

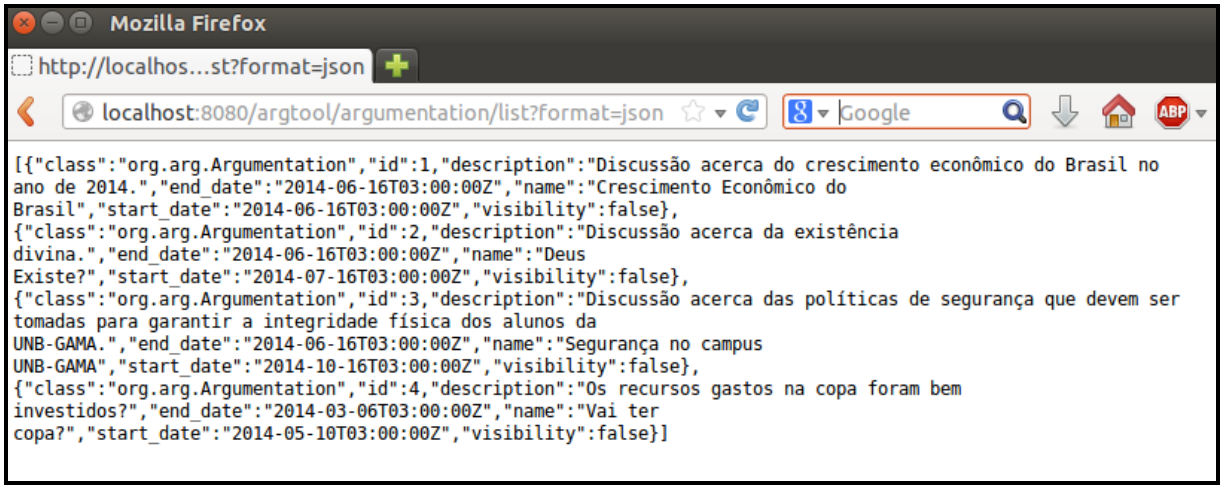

Figura 31 – Serviço RESTful de Listagem das Argumentações

Para acessar um serviço RESTful, basta realizar um requisição HTTP através da URL. Pelo o navegador é possível realizar apenas chamadas do tipo GET. Para testar serviços desenvolvidos a partir de métodos POST, PUT e DELETE, pode-se utilizar ferramentas de transferência de dados como o cURL[2] ou plugins para navegadores como o REST Console do Google Chrome.

Serviços RESTful para a criar (*save*), pesquisar (*show*), deletar (*delete*) e atualizar (*update*) entidades da base de dados também foram desenvolvidos. A seguir, apresentam-se todos os serviços RESTful categorizados a partir das entidades que eles manipulam:

- *Argumentation*: Serviços Implementados: *Save*, *Update*, *Show* e *List*;
- *Node*: Serviços Implementados: *Save*, *Update*, *Show* e *List*. Este serviços podem ser utilizados para a persistência das subclasses de *Node*;
- *User*: Serviços Implementados: *Save*, *Delete*, *List*, *Update* e *Show*.
- *Group*: Serviçõe Implementados: *Save*, *Delete*, *List*, *Update* e *Show*
- *Comment*: Serviçõe Implementados: *Save*, *Delete*, *List*, *Update* e *Show*

5.4 Interface Gráfica do Sistema

A ferramenta ACE-CAST possui um layout padrão reutilizado em todas as páginas. O layout é apresentado na Figura 32.

[2] Disponível em: *http* : *//curl.haxx.se/*

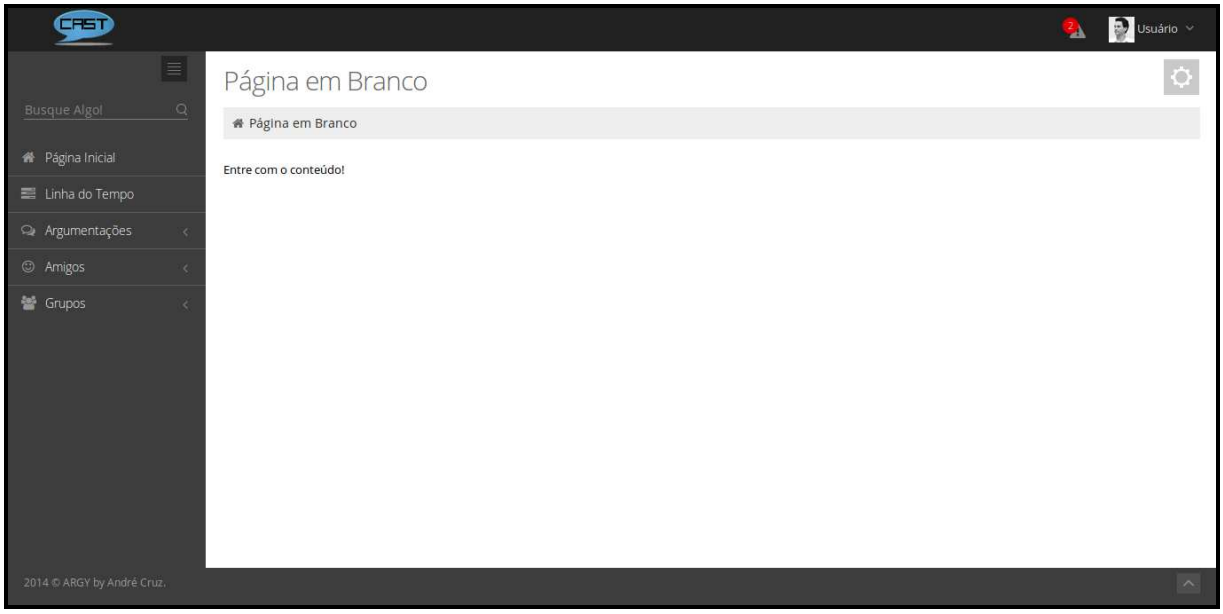

Figura 32 – Layout da Aplicação

Para facilitar a adaptação do usuário, é permitido a customização da interface com base em suas preferências. O botão com formato de uma engrenagem oferece a possibilidade de alterar o tema padrão e customizar a posição dos cabeçalhos, rodapés e menus. Apresenta-se na Figura 33, a customização da interface com o tema azul e o menu lateral no lado direito.

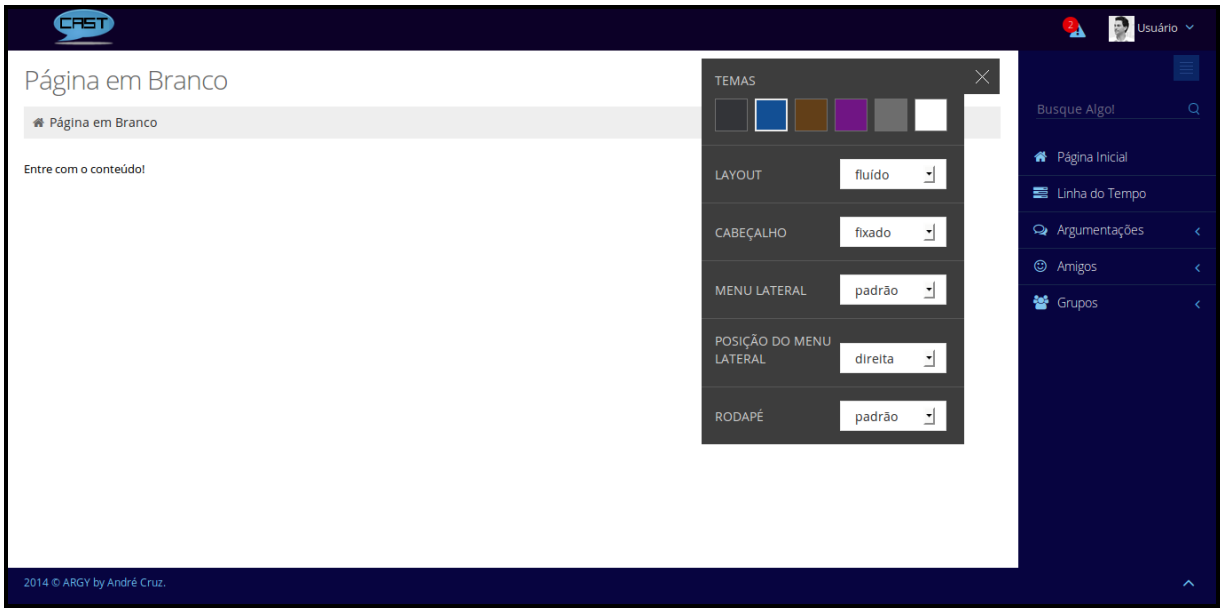

Figura 33 – Customizando o Layout da Aplicação

5.5 Cenário de Uso

Aproximadamente 87% dos usuários que testaram a ferramenta costumam participar em discussões onlines, seja em listas de comentários ou redes sociais. Os resultados obtidos ao término do experimento são apresentados através das Figuras 34, 35, 36, 37, 38.

Os usuários não tiveram dificuldade para criar uma argumentação. Visando facilitar o entendimento alguns usuários sugeriram a mudança da nomenclatura de alguns botões.

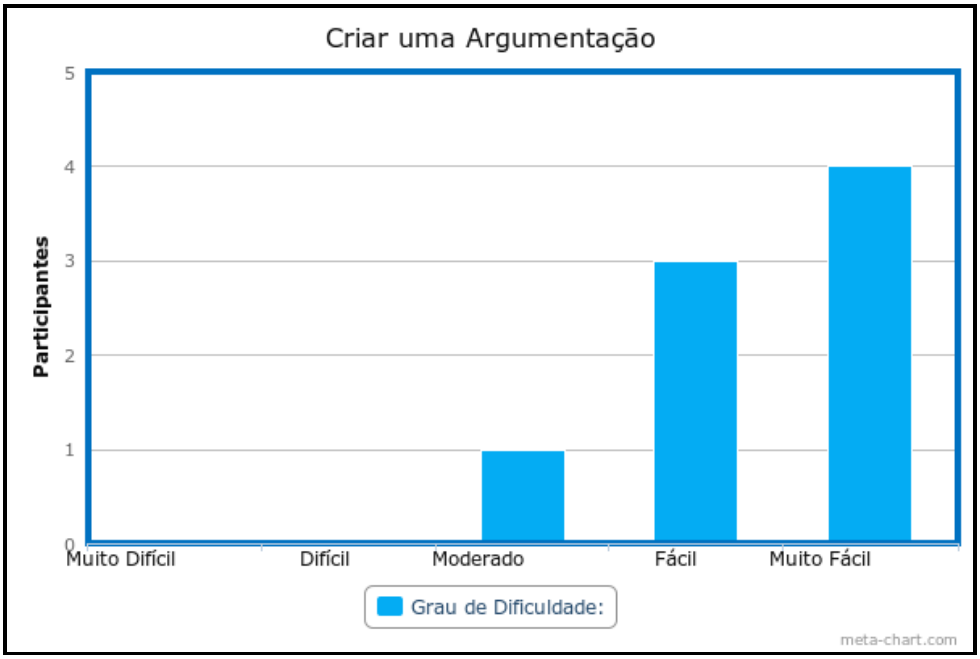

Figura 34 – Criar uma Argumentação

Os usuários acharam fácil adicionar usuários em uma argumentação. Uma questão à ser considerada em trabalhos futuros será a remoção de um usuário de uma argumentação.

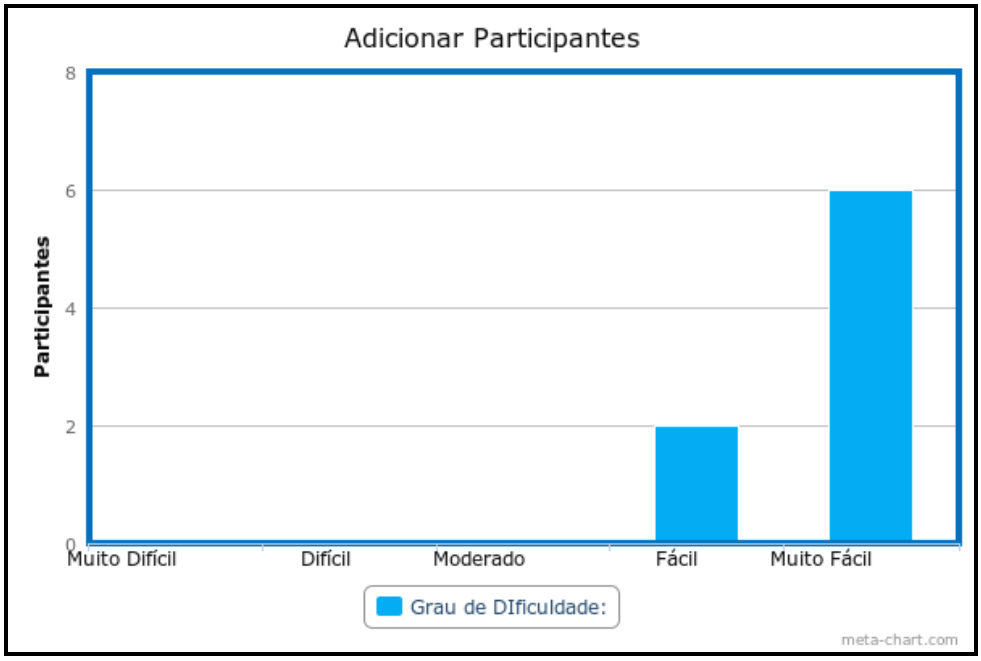

Figura 35 – Adicionar Participantes.

Aproximadamente 75% do usuários conseguiram adicionar novos argumentos à argumentação. A relação de preferência obteve alguma resistência por parte do grupo. Uma preferência só pode ser realizada caso existam dois argumentos conflitantes.

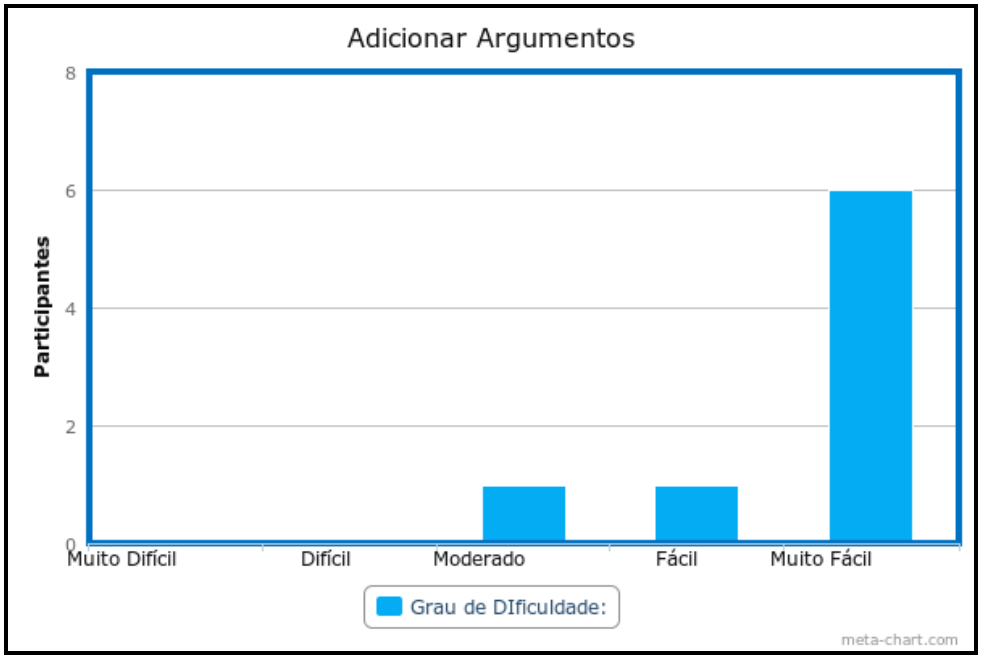

Figura 36 – Adicionar Argumentos.

Todos os usuários conseguiram verificar a validade dos argumentos. O esquema de cores dos *post its* auxiliou neste quesito.

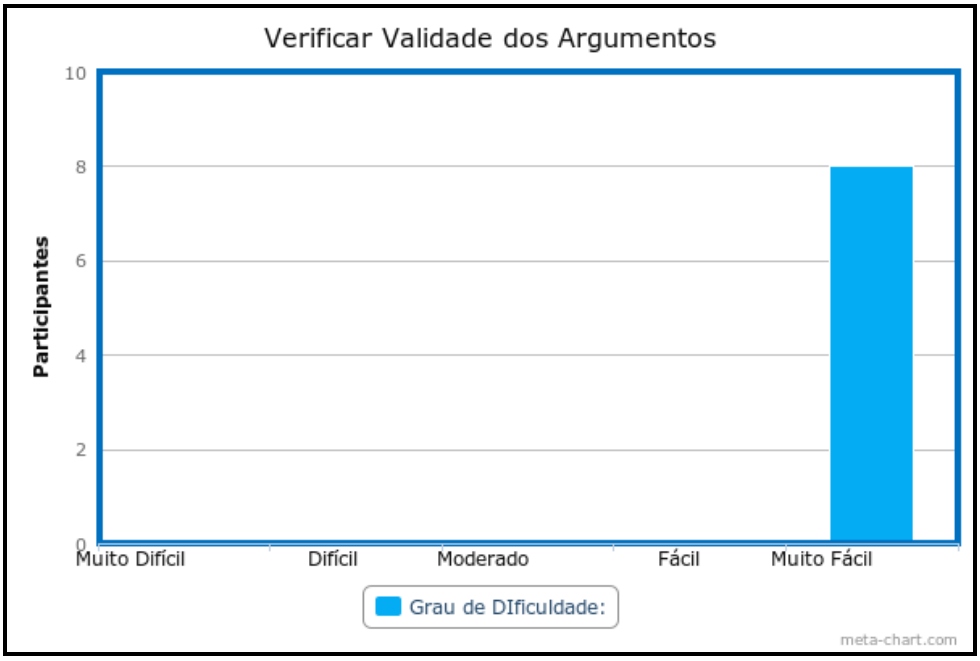

Figura 37 – Verificar Validade dos Argumentos.

A possibilidade de alterar o tema da aplicação foi aprovada pelos usuários. Como cada usuário possui suas preferências, esta funcionalidade permite que o mesmo possa customizar a interface de forma amigável de acordo com suas necessidades.

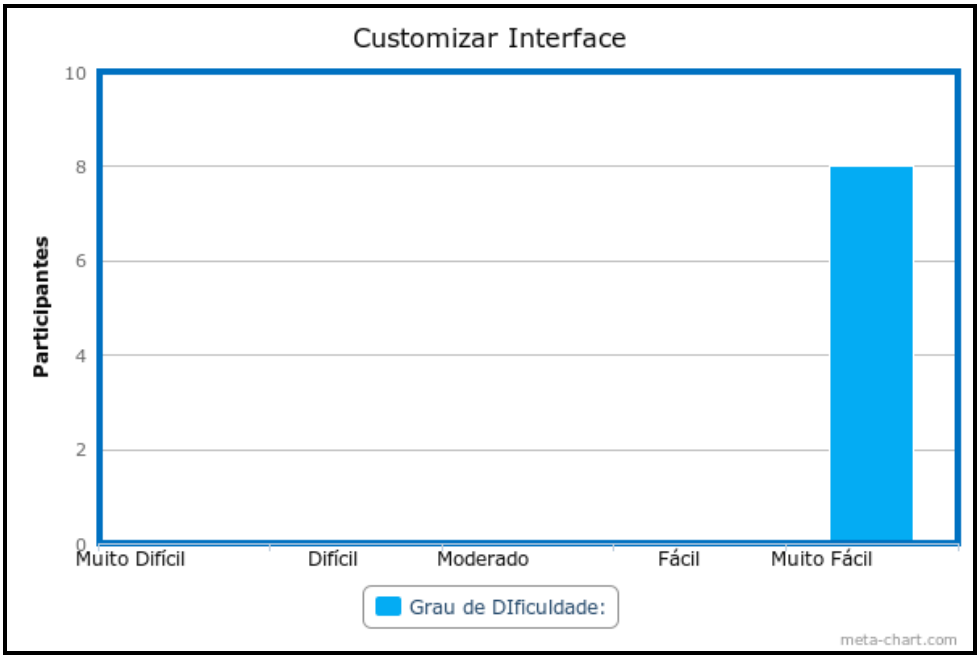

Figura 38 – Customizar Interface.

Ao término do experimento, os usuários consideraram a ferramenta ACE-CAST viável para a condução de uma discussão formal. O comportamento em tempo real do grafo

de argumentação e a possibilidade de rastrear possíveis justificativas para as decisões tomadas fazem com que o usuário tenha facilidade de construir e interagir com novas ideias.

6 Considerações Finais

A argumentação está cada vez mais presente nas atividades colaborativas realizadas pela internet. O papel da argumentação é fazer com que pessoas com pontos de vistas diferentes cheguem a um consenso válido. A definição de um consenso não é uma tarefa trivial. As ferramentas utilizadas atualmente não oferecem suporte completo ao processo de argumentação. Permitindo apenas a diagramação dos elementos de uma discussão.

Em uma discussão é necessário identificar quais argumentos estão aceitos ou não. Abordagens formais baseadas em grafos direcionados que visam enfatizar estes fatores foram propostas. As ferramentas desenvolvidas que seguem estas abordagens são complexas, e geralmente são utilizadas em contextos didáticos. Com este problema em vista, este trabalho tem como objetivo desenvolver uma ferramenta web colaborativa que ofereça suporte às abordagens formais de argumentação. Esta ferramenta deve permitir que diversos usuários possam estruturar suas discussões de forma intuitiva e avaliar a aceitabilidade de cada argumento automaticamente.

A análise de requisitos utilizando técnicas orientadas à meta foi satisfatória. A partir desta abordagem, foi possível avaliar várias soluções diferentes para o mesmo problema. Com base na análise orientada à meta, as histórias de usuário foram definidas sem maiores problemas.

A arquitetura definida, foi elaborada utilizando frameworks de desenvolvimento modernos e eficientes. Esta arquitetura oferece uma solução híbrida que atende os requisitos identificados. A arquitetura sofreu uma modificação no decorrer deste trabalho. Uma das principais premissas da ferramenta a ser desenvolvida é que o grafo de argumentação deve ser atualizado em tempo real. A API REST de persistência foi desenvolvida em Grails e isto representava um gargalo na ferramenta. Pois, o módulo desenvolvido em Node.js suporta um número bem maior de requisições do que o módulo desenvolvido em Grails. Então, a API REST foi incorporada no módulo *Core*. Esta decisão facilitou na manutenção da ferramenta, pois, todos os componentes estão na linguagem JavaScript.

A ferramenta desenvolvida possui um formato inspirado em uma rede social. Os usuários que validaram a ferramenta enfatizaram a facilidade de começar uma argumentação e interagir com outras pessoas. A partir da semântica presente no framework ACE, os usuários identificaram com eficiência os argumentos relacionados com seus argumentos. Com esta informação, os usuários possuem mais liberdade e confiança para adicionar novos argumentos a discussão.

6.1 Trabalhos Futuros

A partir da validação da ferramenta e da experiência do desenvolvedor, um conjunto de itens interessantes podem ser inseridos na ferramenta:

- Criar um sistema de ranking para a ferramenta. A competição sadia entre os usuários é viável no contexto da argumentação. Caso cada argumento ofereça a possibilidade de voto, os usuários irão embasar melhor seus argumentos;
- Explorar outros algoritmos utilizados em grafos na ferramenta ACE-CAST e avaliar os resultados;
- Conduzir testes de validação mais criteriosos para avaliar questões como acessibilidade e demais aspectos associados à usabilidade;
- Como tratar uma possível remoção de usuários da argumentação. O que aconteceria com os argumentos do usuário removido?
- Integrar a ferramenta com o facebook para que os contatos de determinado usuário tenham consciência e possam participar das argumentações que estão sendo realizadas;
- Aplicar a ferramenta em contextos mais fechados. Segundo Walton e Krabbe (1995) e McBurney e Parsons (2001), a discussão pode ser dividida em várias sub áreas. Cada ramo contem um conjunto de regras semânticas e etapas que são aplicadas em determinado domínio. Apresenta-se na Tabela 17 os tipos de argumentação encontrados na pesquisa:

Tabela 17 – Tipos de Argumentação. Fonte: (WALTON, 2010)

Tipo	**Situação**	**Meta dos Participantes**	**Meta do Diálogo**
Persuasão	Conflito de opiniões.	Persuadir a outra parte.	Resolver ou esclarecer uma pendência.
Inquérito	Necessidade de ter uma prova.	Encontrar e verificar evidências.	Confirmar ou refutar hipóteses.
Descoberta	Necessidade de encontrar uma explicação dos fatos.	Encontrar e defender hipóteses convenientes.	Escolher a melhor hipótese para testar.
Negociação	Conflito de interesses.	Conseguir aquilo que é desejado.	Definir um acordo aceitável entre todas as partes.

Continua na Próxima Página

Tabela 17 – *Continuação da Página Anterior*

Tipo	**Situação**	**Meta dos Participantes**	**Meta do Diálogo**
Procura de Informação	Necessidade de Informação.	Adquirir ou oferecer informação.	Extrair informação.
Deliberação	Dilema o escolha prática.	Coordenar objetivos e ações.	Decidir a melhor condução de uma ação.
Eurística	Conflito pessoal.	Verbalmente atacar o oponente.	Revelar premissas mais profundas de um conflito.

Referências

ABDUL-JAWAD, B. *Groovy and Grails Recipes*. Apress, 2009. Disponível em: <http://link.springer.com/content/pdf/10.1007/978-1-4302-1046-7.pdf>. Citado na página 51.

BENOIT, W.; HAMPLE, D. *Readings in Argumentation*. De Gruyter, 1992. (Studies of Argumentation in Pragmatics and Discourse Analysis). ISBN 9783110885651. Disponível em: <http://books.google.com.br/books?id=Qh7Mm45B__UC>. Citado na página 30.

BENTAHAR, J.; MOULIN, B.; BéLANGER, M. A taxonomy of argumentation models used for knowledge representation. v. 33, n. 3, p. 211–259, 2010. ISSN 0269-2821, 1573-7462. Disponível em: <http://link.springer.com/10.1007/s10462-010-9154-1>. Citado 2 vezes nas páginas 28 e 32.

BESNARD, P.; HUNTER, A. *Elements of argumentation*. MIT Press, 2008. ISBN 9780262268400 026226840X 0262026430 9780262026437 9781435643260 1435643267. Disponível em: <http://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=226100>. Citado na página 27.

BEX, F. et al. Implementing the argument web. v. 56, n. 10, p. 66–73, 2013. Disponível em: <http://dl.acm.org/citation.cfm?id=2500891>. Citado na página 27.

BEX, F.; PRAKKEN, H.; REED, C. A formal analysis of the AIF in terms of the ASPIC framework. In: *COMMA*. [s.n.], 2010. p. 99–110. Disponível em: <http://www.researchgate.net/publication/221650992_A_formal_analysis_of_the_AIF_in_terms_of_the_ASPIC_framework/file/9fcfd50c767c99c199.pdf>. Citado 2 vezes nas páginas 42 e 43.

BORGIDA, A. T. et al. (Ed.). *Conceptual Modeling: Foundations and Applications, Essays in Honor of John Mylopoulos*. Berlin: Springer, 2009. (Lecture Notes in Computer Science, v. 5600). ISBN 978-3-642-02462-7. Citado na página 48.

BREWKA, G.; POLBERG, S.; WOLTRAN, S. Generalizations of dung frameworks and their role in formal argumentation. *Intelligent Systems, IEEE*, v. 29, n. 1, p. 30–38, Jan 2014. ISSN 1541-1672. Citado 2 vezes nas páginas 35 e 37.

CHARWAT, G. et al. *Implementing Abstract Argumentation - A Survey*. [S.l.], 2013. Disponível em: <http://www.dbai.tuwien.ac.at/research/report/dbai-tr-2013-82.pdf>. Citado na página 21.

CONKLIN, J.; BEGEMAN, M. L. gIBIS: a hypertext tool for exploratory policy discussion. v. 6, n. 4, p. 303–331, 1998. Disponível em: <http://dl.acm.org/citation.cfm?id=59297>. Citado na página 35.

DANIIL, C.; DASCALU, M.; TRAUSAN-MATU, S. Automatic forum analysis: A thorough method of assessing the importance of posts, discussion threads and of users' involvement. In: *Proceedings of the 2Nd International Conference on Web Intelligence, Mining and Semantics*. New York, NY, USA: ACM, 2012. (WIMS '12),

p. 37:1–37:9. ISBN 978-1-4503-0915-8. Disponível em: <http://doi.acm.org/10.1145/2254129.2254175>. Citado na página 23.

DUNG, P. M. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, v. 77, n. 2, p. 321 – 357, 1995. ISSN 0004-3702. Disponível em: <http://www.sciencedirect.com/science/article/pii/000437029400041X>. Citado 5 vezes nas páginas 22, 23, 29, 35 e 37.

EBADI, T.; PURVIS, M.; PURVIS, M. A. A collaborative web-based issue based information system (IBIS) framework. 2009. Disponível em: <http://otago.ourarchive.ac.nz/handle/10523/1086>. Citado 3 vezes nas páginas 33, 34 e 35.

EEMEREN, F. H. van; GROOTENDORST, R. F.; HENKEMANS, F. S. *Fundamentals of Argumentation Theory: A Handbook of Historical Backgrounds and Contemporary Applications*. Hillsdale NJ, USA: Lawrence Erlbaum Associates, 1996. Citado 2 vezes nas páginas 21 e 28.

GASPER, D. R.; GEORGE, R. V. Analyzing argumentation in planning and public policy: assessing, improving, and transcending the Toulmin model. *Environment and Planning B: Planning and Design*, v. 25, n. 3, p. 367–390, May 1998. Disponível em: <http://ideas.repec.org/a/pio/envirb/v25y1998i3p367-390.html>. Citado na página 30.

GONZÁLEZ, M. P. et al. Developing argument assistant systems from a usability viewpoint. In: *KMIS*. [S.l.: s.n.], 2010. p. 157–163. Citado na página 23.

GUERRERO, L. A.; PINO, J. A. Preparing Decision Meetings at a Large Organization. In: *IFIP WG 8.3 Open Conference on Decision Making and Decision Support in the Internet Age*. [s.n.], 2002. p. 85–95. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.86.1485>. Citado na página 35.

HITCHCOCK, D. Good reasoning on the toulmin model. v. 19, n. 3, p. 373–391, 2005. ISSN 0920-427X, 1572-8374. Disponível em: <http://link.springer.com/10.1007/s10503-005-4422-y>. Citado na página 30.

JAQUEIRA, A. et al. Using i* models to enrich user stories. 2013. Disponível em: <http://ceur-ws.org/Vol-978/paper_10.pdf>. Citado na página 50.

JUDD, C. M.; NUSAIRAT, J. F.; SHINGLER, J. *Node.js the Right Way: Practical, Server-Side JavaScript That Scales*. Springer, 2008. Disponível em: <http://link.springer.com/content/pdf/10.1007/978-1-4302-1046-7.pdf>. Citado 2 vezes nas páginas 52 e 53.

JURETA, I.; MYLOPOULOS, J.; FAULKNER, S. Analysis of multi-party agreement in requirements validation. In: *Proceedings of the 2009 17th IEEE International Requirements Engineering Conference, RE*. Washington, DC, USA: IEEE Computer Society, 2009. (RE '09), p. 57–66. ISBN 978-0-7695-3761-0. Disponível em: <http://dx.doi.org/10.1109/RE.2009.8>. Citado 5 vezes nas páginas 22, 23, 29, 38 e 41.

KARBACH, J. Using toulmin's model of argumentation. *Journal of Teaching Writing*, v. 6, n. 1, 1987. Citado na página 31.

KUNZ, W. et al. *Issues as elements of information systems*. [S.l.], 1970. Citado 3 vezes nas páginas 21, 29 e 35.

LAKATOS, E.; MARCONI, M. D. A. M. *Fundamentos de metodologia científica*. Atlas, 2010. ISBN 9788522457588. Disponível em: <http://books.google.com.br/books?id=Y2WFRAAACAAJ>. Citado na página 66.

LAMSWEERDE, A. V. Goal-oriented requirements engineering: A guided tour. In: *Proceedings of the Fifth IEEE International Symposium on Requirements Engineering*. Washington, DC, USA: IEEE Computer Society, 2001. (RE '01), p. 249–. Disponível em: <http://dl.acm.org/citation.cfm?id=882477.883624>. Citado na página 48.

MCBURNEY, P.; PARSONS, S. Chance discovery using dialectical argumentation. Springer, v. 2253, p. 414–424, 2001. Disponível em: <http://dblp.uni-trier.de/db/conf/jsai/jsai2001.html#McBurneyP01>. Citado na página 92.

MODGIL, S.; PRAKKEN, H. A general account of argumentation with preferences. *Artificial Intelligence*, v. 195, n. 0, p. 361 – 397, 2013. ISSN 0004-3702. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0004370212001361>. Citado na página 37.

MOOR, A. d.; AAKHUS, M. Argumentation support: From technologies to tools. *Commun. ACM*, ACM, New York, NY, USA, v. 49, n. 3, p. 93–98, mar. 2006. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/1118178.1118182>. Citado 3 vezes nas páginas 23, 24 e 27.

NIELSEN, S.; PARSONS, S. A generalization of dung's abstract framework for argumentation: Arguing with sets of attacking arguments. In: MAUDET, N.; PARSONS, S.; RAHWAN, I. (Ed.). *Argumentation in Multi-Agent Systems*. Springer Berlin Heidelberg, 2007, (Lecture Notes in Computer Science, v. 4766). p. 54–73. ISBN 978-3-540-75525-8. Disponível em: <http://dx.doi.org/10.1007/978-3-540-75526-5_4>. Citado na página 37.

PHAM, A.; PHAM, P.-V. *Scrum in action Agile software project management and development*. Course Technology, 2012. ISBN 9781435459144 1435459148. Disponível em: <http://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=354485>. Citado na página 47.

PODLASZEWSKI, M.; CAMINADA, M.; PIGOZZI, G. An implementation of basic argumentation components. In: *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 3*. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2011. (AAMAS '11), p. 1307–1308. ISBN 0-9826571-7-X, 978-0-9826571-7-1. Disponível em: <http://dl.acm.org/citation.cfm?id=2034396.2034539>. Citado na página 23.

PRAKKEN, H. An abstract framework for argumentation with structured arguments. In: *In Information Technology and Lawyers*. [S.l.]: Springer, 2009. p. 61–80. Citado na página 37.

RAHWAN, I. Guest editorial: Argumentation in multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, Kluwer Academic Publishers, v. 11, n. 2, p. 115–125,

2005. ISSN 1387-2532. Disponível em: <http://dx.doi.org/10.1007/s10458-005-3079-0>. Citado na página 27.

RAHWAN, I. Mass argumentation and the semantic web. *Web Semant.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 6, n. 1, p. 29–37, fev. 2008. ISSN 1570-8268. Disponível em: <http://dx.doi.org/10.1016/j.websem.2007.11.007>. Citado na página 22.

RAHWAN, I. et al. Representing and classifying arguments on the semantic web. *The Knowledge Engineering Review*, v. 26, p. 487–511, 12 2011. ISSN 1469-8005. Disponível em: <http://journals.cambridge.org/article_S0269888911000191>. Citado na página 42.

REED, C.; ROWE, G. Araucaria: software for argument analysis, diagramming and representation. *International Journal on Artificial Intelligence Tools*, v. 13, n. 4, p. 961–979, 2004. ISSN 0218-2130. Dc.publisher: World Scientific Publishing. Citado na página 22.

RELVAS, S.; ANTUNES, P. Análise da aplicabilidade dos modelos de argumentação na resolução alternativa de conflitos. In: *1ª Conferência Ibérica de Sistemas e Tecnologias de Informação, Portugal*. [s.n.], 2006. Disponível em: <http://www.di.fc.ul.pt/~paa/papers/cisti-06.pdf>. Citado na página 30.

SCHNEIDER, J.; GROZA, T.; PASSANT, A. A review of argumentation for the social semantic web. In: . Amsterdam, The Netherlands, The Netherlands: IOS Press, 2013. v. 4, n. 2, p. 159–218. Disponível em: <http://dl.acm.org/citation.cfm?id=2590215.2590218>. Citado 2 vezes nas páginas 21 e 22.

SCHWABER, K. *Agile Project Management With Scrum*. Redmond, WA, USA: Microsoft Press, 2004. ISBN 073561993X. Citado na página 45.

SCHWABER, K.; SUTHERLAND, J. *Guia do Scrum: Um guia definitivo para o Scrum: As regras do jogo*. [S.l.], 2013. Disponível em: <https://www.scrum.org/Portals/0/Documents/Scrum%20Guides/2013/Scrum-Guide-Portuguese-BR.pdf>. Citado na página 45.

SMITH, G.; LEDBROOK, P. *Grails in action*. [S.l.]: Manning, 2009. ISBN 9781933988931 1933988932. Citado na página 51.

TOULMIN, S. *The Uses of Argument*. Cambridge University Press, 1958. Disponível em: <http://books.google.com.br/books?id=uffWAAAAMAAJ>. Citado 4 vezes nas páginas 21, 22, 29 e 30.

TRIPP, D. Pesquisa-ação: uma introdução metodológica. Apress, 2005. Disponível em: <http://link.springer.com/content/pdf/10.1007/978-1-4302-1046-7.pdf>. Citado 2 vezes nas páginas 71 e 72.

WALTON, D. Argumentation theory: A very short introduction. In: SIMARI, G.; RAHWAN, I. (Ed.). *Argumentation in Artificial Intelligence*. Springer US, 2009. p. 1–22. ISBN 978-0-387-98196-3. Disponível em: <http://dx.doi.org/10.1007/978-0-387-98197-0_1>. Citado na página 28.

WALTON, D. Types of dialogue and burdens of proof. In: *Proceedings of the 2010 Conference on Computational Models of Argument: Proceedings of COMMA 2010*. Amsterdam, The Netherlands, The Netherlands: IOS Press, 2010. p. 13–24. ISBN 978-1-60750-618-8. Disponível em: <http://dl.acm.org/citation.cfm?id=1860828.1860832>. Citado na página 92.

WALTON, D.; KRABBE, E. Commitment in dialogue. 1995. Citado na página 92.

WILSON, J. R. *Beginning Groovy and Grails*. Pragmatic Bookshelf, 2013. Disponível em: <http://shop.oreilly.com/product/9781937785734.do>. Citado 2 vezes nas páginas 54 e 55.

YU, E. et al. *Social Modeling for Requirements Engineering*. [S.l.]: The MIT Press, 2011. ISBN 0262240556, 9780262240550. Citado na página 48.

YU, E. S.-K. Modelling strategic relationships for process reengineering. University of Toronto, Toronto, Ont., Canada, Canada, 1996. UMI Order No. GAXNN-02887 (Canadian dissertation). Citado na página 48.

ZDRAVKOVIC, J.; SVEE, E.-O.; GIANNOULIS, C. Capturing consumer preferences as requirements for software product lines. *Requirements Engineering*, Springer London, p. 1–20, 2013. ISSN 0947-3602. Disponível em: <http://dx.doi.org/10.1007/s00766-013-0187-2>. Citado na página 50.

ZHANG, W.; LIANG, Y. A temporal based multilateral argumentation dialogue framework. In: *Proceedings of the 2012 International Conference on Computer Application and System Modeling*. Atlantis Press, 2012. Disponível em: <http://www.atlantis-press.com/php/download_paper.php?id=2781>. Citado na página 37.