Framework para o desenvolvimento de aplicações baseadas em modelos de regras adaptativas

F. C. N. Campos, R. F. Feldberg, E. M. M. Jorge, B. M. Trigo

***Resumo*— O objetivo deste trabalho é o desenvolvimento de um framework para o desenvolvimento de aplicações utilizando sistemas de regras adaptativas, visando facilitar a utilização de adaptatividade em aplicações computacionais. Para isto, esta ferramenta se baseia na utilização de tabelas de decisão adaptativas, em conjunto com a simplicidade de descrição do modelo através de uma linguagem baseada em XML. Espera-se, com este trabalho, difundir a utilização de sistemas adaptativos tanto por usuários com pouca experiência no campo da adaptatividade e poucos conhecimentos de programação como por profissionais experientes desejando agilizar o processo de desenvolvimento.**

***Palavras Chave*— Sistemas baseados em regras, Adaptatividade, Tabelas de Decisão, Desenvolvimento, Framework.**

I. Introdução

MECANISMOS adaptativos possuem um largo campo de aplicações em sistemas computacionais, muitos deles ainda inexplorados sob o ponto de vista da adaptatividade. Um dos problemas no qual a plena difusão da tecnologia adaptativa esbarra é a baixa disponibilidade de ferramentas auxiliares, tanto para aprendizado quanto para o desenvolvimento.

Com este trabalho, pretende-se desenvolver um framework para desenvolvimento mais rápido de aplicações utilizando sistemas baseados em regras adaptativas. Para isto, apóia-se na premissa de que o aprendizado e utilização de adaptatividade em aplicações computacionais muitas vezes é desestimulado pela dificuldade do aprendizado de todo o formalismo necessário para o desenvolvimento do motor adaptativo da aplicação, dificultado ainda mais pela ausência de ferramentas que suavizem a curva de aprendizado.

Além de prover apoio a pesquisadores, estudantes e desenvolvedores interessados na tecnologia adaptativa, o framework proposto deve ao mesmo tempo permitir uma grande flexibilidade no que diz respeito aos campos de atuação onde poderia ser aplicado. Com isto, o framework teria não só apelo didático, como poderia ser utilizado para o desenvolvimento de aplicações em escala comercial.

F.C.N. Campos, R.F. Feldberg, E.M.M. Jorge e B.M. Trigo são estudantes de Engenharia Elétrica com ênfase em Computação na Escola Politécnica da Universidade de São Paulo, e podem ser contactados pelo e-mail felipe.pqi@gmail.com, rafael.feldberg@gmail.com, edummjorge@gmail.com e brunomtrigo@gmail.com, respectivamente.

II. Sistemas de Regras e Tabelas de Decisão Adaptativas

Sistemas baseados em regras são, de forma simples, um conjunto de fatos e regras SE-ENTÃO que coordenam todo o funcionamento de um dado sistema. Apesar do conceito simples, sistemas baseados em regras são de suma importância no campo dos sistemas especialistas, uma vez que podem representar de forma simples o raciocínio de um especialista. [4] Além disso, a utilização de bancos de regras e fatos permite uma separação do desenvolvimento da aplicação e das regras que regem seu controle, de forma que modificações no funcionamento do programa possam ser feitas simplesmente alterando o conjunto de regras.

A arquitetura de um sistema simples baseado em regras pode ser vista na figura 1.

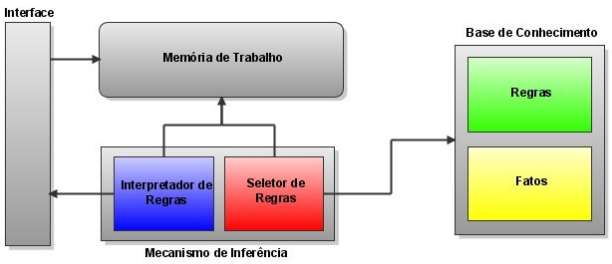


Fig. 1. Arquitetura de um sistema simples baseado em regras.

Através da arquitetura mostrada na figura 1, pode-se ver os principais componentes de um sistema baseado em regras. A memoria de trabalho armazena o estado atual da aplicação, tal qual, valores de variáveis, e quaisquer valores temporários do processamento de regras. A base de conhecimento é um banco onde são armazenadas as regras e fatos que regem o sistema. Este banco é acessado pelo seletor de regras, que seleciona a regra a ser aplicada e a passa ao interpretador de regras, que aplica a regra recebida. O interpretador de regras pode tanto modificar valores da memória de trabalho, como enviar comandos à aplicação através da interface. A aplicação, por sua vez, pode modificar a memória de trabalho como entrada do sistema baseado em regras.

O componente seletor de regras pode seguir uma variedade de estratégias de seleção de regras. No caso de apenas uma regra ser aplicável ao sistema, não há muito sobre o que

decidir. Já no caso de mais de uma regra poder ser aplicada ao estado atual do sistema, estas estratégias, denominadas estratégias de resolução de conflitos, são utilizadas. A estratégia mais simples conhecida é a estratégia da "Primeira Aplicável". Esta estratégia consiste simplesmente em aplicar a primeira regra encontrada. Outras regras comuns são a "Aleatória" (uma das regras aplicáveis é selecionada aleatoriamente), "Mais específica" (a regra que possui maior número de condições cumpridas, dentre as aplicáveis, é selecionada), "Melhor Aplicável" (através da utilização de pesos nas regras, de forma que a regra com maior peso é aplicada), entre diversas outras. Cada uma é melhor aplicada em determinadas situações. Em jogos, por exemplo, a estratégia aleatória é muito útil na inteligência artificial de adversários, devido à imprevisibilidade.

Além disso, existem dois tipos mais conhecidos de algoritmos para sistemas baseados em regras. O primeiro é o algoritmo de Encadeamento à Frente (*Forward-Chaining*) ou Dirigido a Estímulos (*Stimulus-Driven*). Este algoritmo parte do estado do sistema e, buscando as regras que se aplicam a este estado, tentam chegar a uma conclusão ou ação a ser tomada. Já o algoritmo de Encadeamento Reverso (*Backward-Chaining*) ou Dirigido a Objetivos (*Goal-Directed*) busca, a partir de um determinado objetivo, quais regras chegam a ele. Ao encontrar a regra, transforma todas as suas condições em novos objetivos, e assim prossegue até encontrar um conjunto de fatos e determinar a seqüência de regras para se atingir o objetivo inicialmente proposto.

Já a base de conhecimento pode ser definida através de uma diversidade de formalismos. Um deles são as tabelas de decisão, que são estruturas tabulares para a representação da lógica de tomada de decisão de uma aplicação. Tabelas de decisão são uma forma de mostrar toda a lógica de funcionamento de uma aplicação de forma simples, compacta e fácil leitura. Além disso, são meios eficientes de representação tanto para pessoas dentro da área de computação como fora, facilitando a comunicação entre os desenvolvedores e especialistas e não limitando a sua utilização à área da computação. Outra vantagem trazida pela utilização de tabelas de decisão é que estas podem ser traduzidas diretamente para lógica computacional, sem necessidade de tradução. [6]

Tabelas de decisão são representadas basicamente através de quatro quadrantes, conforme mostra a tabela 1.

TABELA I
ESTRUTURA BÁSICA DE UMA TABELA DE DECISÃO

Linhas de Condição	Entradas de Condição
Linhas de Ação	Entradas de Ação

Note que a tabela de decisão exemplificada possui as linhas de condição separadas das linhas de ação através de uma linha dupla: este padrão é adotado para facilitar a leitura da tabela.

Na tabela, cada coluna representa uma regra. Assim, as linhas de condição expressam qual a condição sendo testada, e as entradas de condição especificam os valores de cada condição para o qual aquela regra é aplicada. Já as linhas de ação especificam quais as ações a serem tomadas, e as entradas de ação especificam parâmetros ou se aquela ação deve ser executada quando a regra é aplicada. Um exemplo de tabela de decisão pode ser vista na tabela 2.

TABELA 2
EXEMPLO DE TABELA DE DECISÃO

	Regra 1	Regra 2	Regra 3
Crédito	SATISFATÓRIO	INSUFICIENTE	INSUFICIENTE
Histórico		BOM	RUIM
Compra	ACEITAR	REJEITAR	REJEITAR

Poderíamos adicionar às tabelas de decisão previamente mostradas um módulo adaptativo subjacente. Desta forma, poderíamos ter sistemas baseados em regras adaptativas, com regras mutáveis durante a execução do sistema sem interferência externa. Assim, a estrutura básica de uma tabela de decisão adaptativa poderia ser representada conforme a tabela 3.

TABELA 3
ESTRUTURA BÁSICA DE UMA TABELA DE DECISÃO ADAPTATIVA

Linhas de Condição	Entradas de Condição
Linhas de Ação	Entradas de Ação
Funções Adaptativas "before-"	Entradas de Função
Funções Adaptativas "after-"	Entradas de Função

Desta forma, as linhas de funções adaptativas "before-" definem as funções adaptativas que são executadas antes da aplicação da regra, e as linhas de funções adaptativas "after-" definem as funções adaptativas que são executadas após a aplicação da regra. As entradas de função definem quais funções adaptativas são chamadas e com quais valores de parâmetros.

Uma função adaptativa é definida através de um conjunto de ações adaptativas, que podem adicionar regras, remover regras, entre outros. Assim, um exemplo de tabela de decisão adaptativa pode ser vista na tabela 4.

TABELA 4
EXEMPLO DE TABELA DE DECISÃO ADAPTATIVA

	H	-	-	+	+	+	Regra 1	Regra 2
Estado =		p_1	p_1	p_1	g_1	p_1	1	1
Entrada =		*a*	*b*	*b*	*b*	*a*	*a*	*b*
Estado :=		p_1	p_2	g_1	p_2	p_1	1	2
Função A	A	✓				✓	✓	
p_1	P	p_1				p_1	1	
p_2	P	p_2				p_2	2	
g_1	G							

Note no exemplo dado na tabela 4 que antes das colunas de regra vêm as colunas que definem as funções adaptativas. A coluna H define o cabeçalho de uma função adaptativa: o valor identificado após o nome da função indica se é uma função "after-" (A) ou "before-" (B). Abaixo da função vêm seus parâmetros (P) e geradores (G). As colunas subseqüentes identificam as ações adaptativas que fazem parte daquela função, sendo um (-) para remoção de regra e um (+) para adição de regras. Os valores destas colunas nas entradas de condição e ação definem os valores da regra sendo adicionada ou removida, e os valores nas linhas de funções adaptativas indicam as chamadas e valores de parâmetros das regras sendo adicionadas ou removidas. Os valores das linhas de funções nas colunas de regras identificam se a função adaptativa é chamada, e dão os valores para cada parâmetro para esta chamada.

III. DESENVOLVIMENTO DO FRAMEWORK

Com base nos objetivos e nos conceitos explorados anteriormente, foi definida a arquitetura básica para o framework demonstrada na figura 2.

Fig. 2. Arquitetura do framework adaptativo.

O framework seria executado internamente à aplicação, sendo responsável por todo o controle de execução do modelo. Como pode ser visto pelo diagrama da arquitetura, todo o framework utiliza uma tabela de decisão como fundação para seu funcionamento. Tal escolha se deu, inicialmente, pelo fato que tabelas de decisão incorporam as principais idéias de um sistema básico baseado em regras. [7] Além disso, a estrutura simples e tabular das tabelas de decisão facilitaria a modelagem do sistema por usuários menos experientes. Outra grande vantagem na utilização de tabelas de decisão adaptativas é a relativa simplicidade para a modelagem de outros dispositivos adaptativos utilizando a mesma, quando comparado a outros formalismos. Uma tabela de decisão permite que novos dispositivos adaptativos sejam modelados através de manipulações simples e diretas de variáveis, ou seja, sem necessidade de conversões trabalhosas e dispendiosas do ponto de vista de recursos computacionais.

A tabela de decisão utilizada pelo framework foi modelada seguindo o conceito apresentado, porém algumas modificações foram feitas para que houvesse uma maior flexibilidade e facilidade de modelagem. Uma destas modificações é a presença da variável de "Estado" já embutida no sistema, como pode ser visto pela presença de um módulo dedicado aos estados da aplicação. Outra modificação diz respeito ao módulo de funções adaptativas. Na teoria formal, as funções adaptativas são definidas previamente como sendo do tipo "after-" ou "before-". Neste projeto, as funções não mais são definidas previamente, mas podem ser chamadas a qualquer momento da aplicação da regra, seja antes, depois ou até mesmo ambos. Com isso, espera-se ganhar em flexibilidade e facilidade de modelagem.

A Tabela de Regras é o módulo responsável por armazenar todas as regras atuais do modelo. Este módulo é alterado pelas ações adaptativas de cada função presente no módulo de Funções Adaptativas quando são chamadas. Este módulo foi desenvolvido visando um sistema baseado por regras com Encadeamento à Frente, ou seja, buscando uma ação de acordo com o estado atual do sistema. A busca de regras utiliza, como estratégia de resolução de conflitos, o método de "Primeira Aplicável". Tal decisão foi tomada visando facilitar a modelagem dos sistemas, por ser mais simples, e também por possuir melhor desempenho em comparação às demais. A forma modular como foi desenvolvido o framework permite que, mais tarde, outras estratégias de resolução de conflito sejam adicionadas.

Pode-se notar que existem dois módulos de variáveis: as variáveis da tabela e as variáveis do modelo. Isto foi feito pois uma das funcionalidades do framework é prover, além das tabelas de decisão adaptativas mencionadas, outros dispositivos adaptativos prontos para uso mais simples. Desta forma, caso um dispositivo adaptativo diferente de uma tabela de decisão esteja em uso, as variáveis do modelo são utilizadas pelo usuário para definir o estado do sistema e então são convertidas pelo framework nas variáveis da tabela, utilizadas para modelar o dispositivo adaptativo escolhido pelo usuário. Além disso, os métodos do modelo trazem métodos de nível mais alto para facilitar a interação do usuário com o framework. Por exemplo, a tabela de decisão adaptativa só permite que o usuário execute um passo da tabela por vez através do método *runStep()*, uma vez que não se pode saber como é o funcionamento do modelo previamente. Caso o usuário esteja utilizando um Autômato Adaptativo Finito, no caso um dispositivo já disponibilizado pelo framework, este dispõe de um método adicional que permite que seu modelo seja executado inteiramente para uma dada cadeia de entrada, verificando se a cadeia é ou não aceita

pelo sistema. Sendo assim, estes módulos visam facilitar a utilização do framework de forma mais direta e prática por aqueles que não possuem tanto conhecimento ou desejam desenvolver algo mais rapidamente.

Outro módulo bastante útil adicionado ao framework é a Interface. Este módulo dá ao usuário uma interface que pode ser implementada por sua aplicação, permitindo uma comunicação mais próxima entre a aplicação e o modelo. A interface permite que o modelo, durante a execução de uma regra, chame um método da aplicação onde reside, seja para avisar de um evento ou para requisitar algum tratamento adicional. Esta chamada de método é feita como uma ação adaptativa, e pode enviar para a aplicação dois parâmetros. Desta forma, o modelo possui um tipo de chamada que funciona quase como um tipo de interrupção à aplicação, passando parâmetros do modelo e permitindo um controle maior sobre o fluxo do mesmo.

IV. Linguagem de Descrição de Modelos

Um dos problemas encontrados durante testes e desenvolvimento de provas de conceito utilizando o framework foi a dificuldade em ler e entender o modelo através do código-fonte da aplicação, onde este era construído inteiramente através de métodos do framework. Apesar de possuir métodos de alto nível voltados para uma construção mais simples do modelo, a leitura do modelo através dos parâmetros dos métodos não era eficaz quando este modelo crescia em complexidade, e assim surgiu uma necessidade de criar uma segunda forma de criar o modelo na aplicação.

Para atender a esta necessidade, foi criada uma linguagem baseada em XML para descrever o modelo em utilização. O arquivo contendo esta descrição pode então ser importado pela aplicação durante a execução para carregar o modelo. A escolha de uma linguagem baseada em XML se deu, inicialmente, pela estrutura em árvore da mesma, o que permite uma melhor visualização da hierarquia do modelo, além de ser de fácil aprendizagem e de simples leitura.

A utilização de um arquivo de descrição do modelo trouxe outras vantagens, como o fato de que o modelo agora pode ter toda a sua lógica de execução modificada sem necessidade de modificar a aplicação em que reside. O arquivo de descrição, por ser em uma linguagem diferente daquela em que a aplicação é escrita, também permite que outras pessoas envolvidas com o projeto, mas não necessariamente programadores, leiam, entendam e modifiquem o modelo, facilitando a comunicação e interação.

Uma vez que o framework já era capaz de importar os modelos através da linguagem de descrição, decidiu-se também adicionar um módulo para exportação de modelos em execução para um arquivo de descrição. Desta forma, se torna mais fácil realizar uma varredura por problemas de modelagem, verificar estados intermediários do modelo e até mesmo realizar a persistência de modelos em execução. Esta possibilidade permite que o framework deste trabalho seja utilizado também em aplicações onde é desejável persistir o estado atual do modelo, o que é especialmente útil em aplicações que utilizam aprendizado e treinamento. Desta forma, aumentou-se a abrangência e campos de aplicação do framework.

V. Implementação e Testes Executados

O framework foi desenvolvido, ao fim, seguindo o diagrama de classes simplificado mostrado na figura 3.

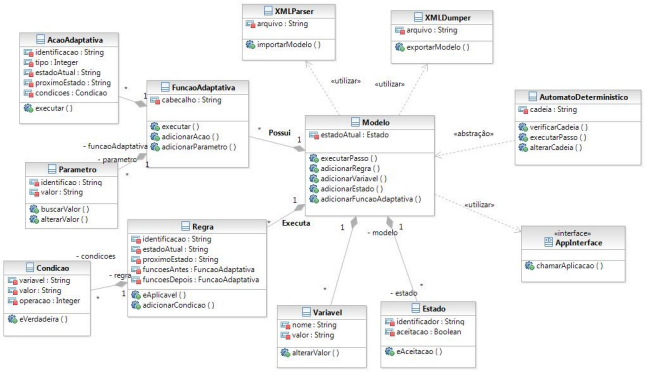


Fig. 3. Diagrama de classes simplificado do framework adaptativo.

A linguagem escolhida para desenvolvimento do framework foi Java. Esta decisão tomou por base não só o fato de ser uma linguagem popular, mas também por, devido a ser executada sobre uma máquina virtual, permitiria que o framework fosse utilizado sobre diversas plataformas diferentes. Assim, podemos citar, entre os diversos campos de aplicação, a possibilidade de utilização do framework para desenvolver aplicações adaptativas para celulares e outros dispositivos móveis, aplicações Web, conteúdo interativo adaptativo para discos Blu-ray (BD-J), além das aplicações tradicionais para computadores pessoais.

Os testes executados tomaram como base o autômato adaptativo que resolve a linguagem $L = \{a^{2n}b^{2n}e \mid n \geq 0\} \cup \{a^{2n+1}b^{2n+1}o \mid n \geq 0\}$, mostrado graficamente na figura 4. Este autômato foi selecionado pois utiliza, para sua execução, todas as funções do framework, tais quais gerador de novos estados, criação e remoção de regras, entre outros.

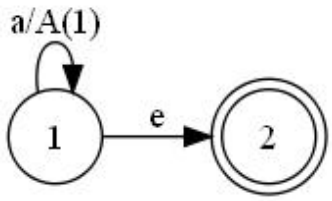


Fig. 4. Autômato utilizado para testes do framework adaptativo.

Tal autômato, ao receber o primeiro caracter "*a*" da cadeia de entrada, executa a função adaptativa A que remove a ligação entre o estado de aceitação 2 e o estado dado como parâmetro p_1 para o símbolo "*e*", remove a ligação entre o estado 1 e ele mesmo, cria um novo estado entre os estados 2 e o estado dado como parâmetro e cria ligações entre o estado do parâmetro p_1 e o novo estado g_1 para o símbolo "*b*", entre o estado g_1 e o estado de aceitação 2 para o símbolo "*o*" e entre o estado 1 e ele mesmo para o símbolo "*a*", chamando a função adaptativa B com o estado criado por g_1 como parâmetro. Dessa forma, o autômato da figura 4, ao receber o símbolo "*a*", se modificaria para o autômato visto na figura 5.

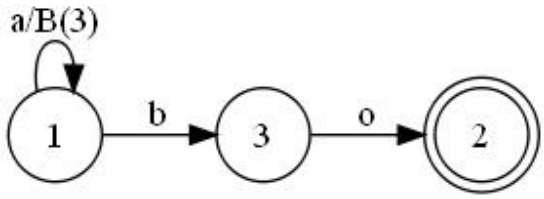


Fig. 5. Autômato utilizado para testes do framework adaptativo após consumir um símbolo "a".

A função adaptativa B, por sua vez, realiza as mesmas ações que a função adaptativa A, porém invertendo o símbolo "*e*" por "*o*" e vice-versa. Além disso, foi utilizada uma ação adaptativa adicional em cada função adaptativa para chamar um método da aplicação de testes que realizava a exportação do modelo para cada passo, para que fosse verificado seu funcionamento.

VI. Conclusões

O desenvolvimento deste framework trouxe uma importante contribuição no preenchimento da lacuna existente hoje na área de tecnologia adaptativa no que diz respeito a ferramentas de auxílio de desenvolvimento e aprendizagem. A definição de uma nova ferramenta que permita o desenvolvimento de aplicações adaptativas sem necessidade de conhecer previamente todo o formalismo de dispositivos regidos a regras adaptativas permitiria a difusão deste novo campo de conhecimento através do estímulo a pesquisadores, estudantes e desenvolvedores, tornando a curva de aprendizado mais suave para aqueles que ainda não estão familiarizados.

Além disso, a utilização de uma linguagem de descrição simples permite que seja utilizado até mesmo por programadores iniciantes ou inexperientes, e ao mesmo tempo provendo uma ferramenta poderosa para programadores experientes que podem desenvolver aplicações de forma simples e rápida utilizando um framework de fácil utilização e ao mesmo tempo bastante flexível.

Assim, esperamos que o projeto aqui desenvolvido incentive e colabore com o desenvolvimento de novas aplicações no campo da tecnologia adaptativa.

Referências

[1] F. C. N. Campos, E. M. M. Jorge, R. F. Feldberg e B. M. Trigo, "*Framework Adaptativo*", Website do projeto. Disponível em: http://www.edu.poli.usp.br/felipe/adapdk. Acesso em: 19 dez. 2009.

[2] M. V. M. Ramos, J. J. Neto, I. S. Vega, "*Linguagens formais: Teoria, Modelagem e Implementação*", Porto Alegre, Bookman, 2009.

[3] J. Freeman-Hargis, "*Rule-based Systems and Identification Trees*". Disponível: http://ai-depot.com/Tutorial/RuleBased.html. Acesso em: 14 Jun. 2009.

[4] F. Hayes-Roth, "Rule-based systems," *Communications of the ACM*, vol. 28, no. 9, pp. 921-932, Sept. 1985.

[5] H. W. Kirk, "Use of decisión tables in computer programming," *Communications of the ACM*, vol. 8, no. 1, pp. 41-43, Jan. 1965.

[6] U. W. Pooch, "Translation of Decision Tables," *Computing Surveys*, vol. 6, no. 2, pp. 125-151, June 1974.

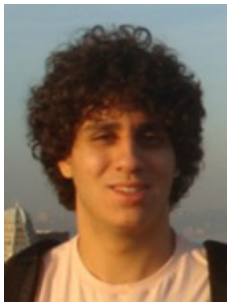
[7] A. Ligeza, "*Logical foundations for rule-based systems*," 2nd ed., Springer, 2006.

[8] J. J. Neto, "Adaptive rule-driven devices – general formulation and case study," *CIAA'01: Revised Papers from the 6th International Conference on Implementation and Application of Automata*, London, UK: Springer-Verlag, 2002, pp. 234-250, ISBN 3-540-00400-9.

[9] H. Pistori, "*Tecnologia adaptativa em engenharia de computação: estado da arte e aplicações*," Dissertação de doutorado, orientada por J. J. Neto, Escola Politécnica, Universidade de São Paulo, 2003.

Felipe C. N. Campos nasceu em Juiz de Fora/MG, Brasil, em 07 de outubro de 1986. É formado em Engenharia Elétrica com ênfase em Computação na Escola Politécnica da Universidade de São Paulo.

Realizou iniciação científica na área de sistemas em Lógica Fuzzy pelo KNOMA (Laboratório de Engenharia do Conhecimento) e trabalha na IBM na área de consultoria em Business Intelligence. Já realizou projetos relacionados à utilização de recursos auxiliares e Robótica no ensino de fundamentos de computação para Engenharia na Escola Politécnica da Universidade de São Paulo, resultando no laboratório de Robótica utilizado hoje como atividade complementar para alunos dos primeiros anos, apresentados em conferência internacional de ensino em engenharia.



Rafael F. Feldberg nasceu em São Paulo/SP, Brasil, em 14 de Dezembro de 1986. É formado em Engenharia Elétrica com ênfase em Computação na Escola Politécnica da Universidade de São Paulo e aluno de graduação da Faculdade de Direito da Universidade de São Paulo.

Trabalhou na Oracle do Brasil na área de Pré-vendas de tecnologia (produtos relacionados a banco de dados e middleware). Na empresa realizou projetos relacionados com segurança da informação, principalmente com gestão de identidade.

Eduardo M. M. Jorge é formado em Engenharia Elétrica com ênfase em Computação na Escola Politécnica da Universidade de São Paulo. Realizou iniciação científica com o desenvolvimento de software para computador de mão voltado à pesquisa agrícola pelo LAA (Laboratório de Automação Agrícola) e trabalha na área de desenvolvimento do Credit Suisse.

Bruno M. Trigo é formado em Engenharia Elétrica com ênfase em Computação na Escola Politécnica da Universidade de São Paulo. Já realizou diversos projetos de pesquisa na área de didática para o ensino de Engenharia, com artigos publicados em congressos no exterior sobre a utilização de Applets no ensino de Química.