Adaptalker: Um Framework para o Gerenciamento Adaptativo do Diálogo

D.A. Alfenas, C. E. Bogik, D. P. Shibata, R. P. da Silva, M. R. Pereira-Barretto

***Abstract*— Dialog management is the central activity of a spoken dialog system, whose responsibility is choosing the communicative actionssent to the system user. This paper presents a formal model for such activity that uses adaptive technology. This model is being used to drive the development of a spoken dialog system framework.**

***Keywords*—spoken dialog systems,dialog management, adaptive technology.**

I. Introdução

Os sistemas de diálogo falado são projetados para o uso da voz como principal canal de interação com o usuário através de linguagem natural. Na maioria desses sistemas, o gerenciador de diálogo é o componente central na arquitetura [1]. Este trabalho apresenta um modelo para o gerenciamento adaptativo do diálogo, primeiramente descrito em [1], e também uma plataforma para construção de gerenciadores, ainda em desenvolvimento, chamada de Adaptalker. Tanto o modelo quanto a plataforma são evoluções de trabalhos apresentados no VI Workshop de Tecnologia Adaptativa [2] e no VII Workshop de Tecnologia Adaptativa [3].

A adaptatividade é um termo que se refere à capacidade de um sistema de tomar a decisão de modificar seu próprio comportamento, em resposta ao seu histórico de operação e aos dados de entradasem a interferência de qualquer agente externo[4]. A utilização de tecnologia adaptativa foi motivadapela sua capacidade de transformar dispositivos livres de contexto em dispositivos sensíveis ao contexto com poucas modificações estruturais. Esta capacidade foi utilizada para aperfeiçoar uma técnica existente e comercialmente estabelecida, o gerenciamento de diálogo baseado em estados finitos, de forma a eliminar ou reduzir as deficiências de tal tecnologia, já discutidas por outros autores [5][6].

O restante deste artigo está estruturado da seguinte maneira: o tópico II apresenta uma revisão sucinta da literatura de gerenciamento de diálogo falado. No tópico III é apresentado, em alto nível, a proposta de uma arquitetura completa para sistema de diálogo falado, no qual a plataforma proposta de gerenciamento está inserida. O tópico IV detalha o modelo de adaptativo de gerenciamento, com foco na utilização da tecnologia adaptativa. O tópico V apresenta a plataforma de desenvolvimento.O último capítulo contém a conclusão sobre o trabalho.

D. A. Alfenas, Fundação para o Desenvolvimento da Engenharia (FDTE), São Paulo, Brasil, d.alfenas@fdte.org.br.

C. E. Bogik, Fundação para o Desenvolvimento da Engenharia (FDTE), São Paulo, Brasil, carlos.bogik@fdte.org.br

D. P. Shibata, Fundação para o Desenvolvimento da Engenharia (FDTE), São Paulo, Brasil, d.shibata@fdte.org.br

R. P. da Silva, Fundação para o Desenvolvimento da Engenharia (FDTE), São Paulo, Brasil, raphael.pinheiro@fdte.org.br

M. R. Pereira-Barretto, Escola Politécnica da USP (EPUSP), São Paulo, Brasil, marcos.barretto@poli.usp.br.

II. Gerenciamento de diálogo e os sistemas de diálogo falado

A. Sobre o diálogo

Os sistemas computacionais de diálogo são baseados no sistema de tomada de turnos para a conversação proposta porSacks, Schegloff e Jefferson [7]. Este sistema determina, entre várias outras regras,que (i) a pessoa que fala muda ao menos uma vez, (ii) que as pessoas falam uma de cada vez, (iii) sobreposições de fala acontecem, mas tendem a ser curtas, (iv) há pouca latência entre os turnos e (v)mecanismos de reparo existem para lidar com erros na tomada de turnos. Os turnos são compostos por unidades menores, chamadas de unidades de construção de turno. Estas unidades podem ser de vários tipos, tais como sílabas, palavras e frases.

A unidade de construção de turno mais comum em sistemas computacionais é o *segmento funcional*. Um segmento funcional pode ser entendido como o menor trecho do turno que tenha uma função comunicativa.Esta últimapode ser definida como o tipo da intenção comunicativa expressa no segmento funcional. A pergunta, a pergunta de verificação (uma especialização da função pergunta), a oferta,o cumprimento e o*feedback* são exemplos de funções comunicativas. Um segmento funcional pode ter mais de uma função comunicativa. Neste caso, diz-se que o segmento tem dois ou mais *atos dialogais*, cada um em uma diferente *dimensão comunicativa* [8]. Um ato dialogal pode ser entendido como uma operação de atualização de estado da informação em cada um dos participantes do diálogo. Ele é definido por uma função comunicativa, um ou mais *itens semânticos* e uma ou mais relações funcionais com outros atos dialogais.Atos dialogais, segmentos funcionais e dimensões comunicativas fazem parte da norma ISO24617-2 para anotação de diálogos [9]. Esta norma define também um conjunto padrão de funções comunicativas.

O seguinte trecho de diálogo descreve um segmento com dois atos dialogais na ação do sistema:

```
Usuário: A que horas sai o próximo trem
para Jundiaí?
Sistema: O próximo trem para Jundiaí
parte de São Paulo 19:35h.
```

Nele, pode-se observar que o turno do sistema tem um ato dialogal de função comunicativa *resposta* (de propósito geral, sem dimensão específica), com item semântico igual ao horário de partida do trem (19:35h) e relação funcional com o ato dialogal de função *pergunta* do turno do usuário. Ele também tem um ato de função*feedback*(específico da dimensão *feedback*), pois repete o conteúdo da pergunta (“o próximo trem para Jundiaí”).

Outro conceito importante do diálogo é o*common ground*, conjunto de informações que um participante supõe que seu interlocutor saiba e que são básicas para o bom relacionamento e funcionamento da conversa[10][11]. *Grounding* é o nome dado ao processo interativo pelo qual o common ground é construído e mantido entre dois ou mais indivíduos[12]. O feedback é o mecanismo do diálogo utilizado para sinalizar o *grounding*.

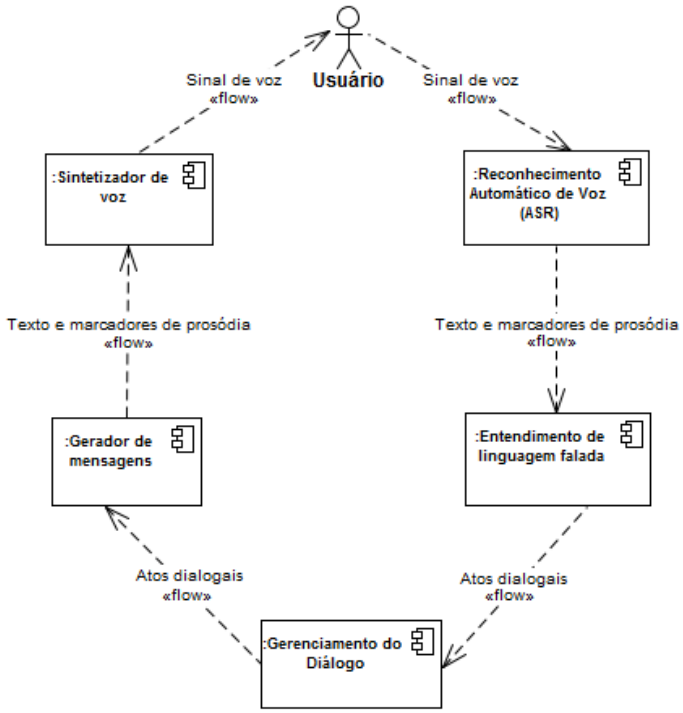

Figura 1: Arquitetura típica de um Sistema de Diálogo Falado

B. *Sobre sistemas computacionais de diálogo falado*

A Figura 1 mostra a arquitetura típica de um sistema de dialogo falado. O reconhecedor de voz é o componente responsável por transformar os sinais de voz em texto e outras anotações relacionadas à prosódia. Entretanto, a prosódia e a parte verbal podem ser tratadas separadamente, como na arquitetura proposta em [2]. O entendimento de linguagem falada é o responsável por geraros segmentos funcionais, atos dialogais e itens semânticos a partir do texto anotado. O gerenciamento de diálogo é um processo que envolve, tipicamente, duas tarefas: uma estimativa do estado corrente do diálogo e a tomada de decisão da próxima ação comunicativa do sistema. Alguns trabalhos deixam explícita a separação entre essas duas tarefas, como em [13]. A estimativa do estado corrente do sistema envolve observar a última ação comunicativa do usuário e relacioná-la com as ações anteriores do sistema e do usuário. O gerador de mensagens transforma a saída do gerenciador de diálogo, formada por atos dialogais ou outros formatos estruturados, em texto em linguagem natural e anotações de prosódia que são, finalmente, transformados em voz pelo sintetizador de voz.

Existem outros componentes, comuns a todo o ciclo de processamento, relacionados à fonte de conhecimento e de dados. Podemos citar, por exemplo, a base de dados léxica (com substantivos, adjetivos, sinônimos, etc.) e a memória semântica. Esta últimacontém as*crenças*do sistema sobre o mundo, incluindo informações sobre o usuário, como identificação e objetivos.

Sistemas *multimodais* possuem módulos específicos para cuidar de outros canais de comunicação que não a voz, tais como expressões faciais e gestos manuais. Eles possuem, tipicamente, um módulo adicional responsável pela *fusão* das informações fornecidas pelos diversos canais de entrada, de forma que eles fiquem devidamente sincronizados. Este tipo de arquitetura é discutido em [14].

Devido a taxas de erros significativas tanto no módulo de reconhecimento quanto no de entendimento de linguagem falada[15], esses módulos não geram, usualmente, saídas únicas sobre o turno do usuário: eles geram um conjunto de hipóteses. Cada hipótese possui uma representação da ação do usuário e um grau de confiança associado. Um gerenciador de diálogo se torna mais robusto ao considerar essas múltiplas hipóteses no processo de tomada de decisão [16].

O sistema de diálogo pode ser classificado quanto à sua capacidade de suportar o processamento *incremental*, isto é, sua capacidade de processar o turno do diálogo enquanto este ainda está sendo falado, ao invés de processar apenas o turno do usuário de uma única vez. A maioria dos sistemas de diálogo pesquisada não são incrementais. Exemplos de arquitetura incrementais podem ser encontrados em [17] e [18].

C. *Sobre as técnicas de gerenciamento de diálogo*

Vários textos fazem um revisão ampla das técnicas de gerenciamento de diálogo e as classificam em grupos, tais como [5], [6] e [19].A técnica em que o Adaptalker se baseia é o gerenciamento baseado em estados finitos. Elafoi e talvez ainda sejaa mais utilizada em sistemas comerciais. Neles, o usuário é levado a uma sequência de passos pré-definidos. O fluxo é especificado como um conjunto de estados ligados por transições denotando os caminhos alternativos conforme as respostas do usuário às solicitações feitas pelo sistema.A maior vantagem deste método é que o vocabulário e gramática podem ser especificados previamente, tornando mais simples o desenvolvimento, treino e teste do sistema. O maior problema é que ele dificulta a modelagem de situações em que o usuário obtém a iniciativa, isto é, situações em que o usuário faz as perguntas ou introduz um assunto. Outro problema é a impossibilidade de construir fluxos dependentes de contexto; por exemplo, não há maneira imediata de o usuárioreparar uma informação dada vários turnos antes, requerendo que ele solicite ao sistema retornar ao passo anterior por várias vezes até atingir o ponto em que a informação errada foi dada,

exceto se houver replicação a cada estado da transição que permite a correção da informação.

Outras técnicas bastante utilizadas incluemgerenciadores baseados em *frames* ou *templates*, baseados em processo de decisão de Markov ou emprocesso de decisão de Markov parcialmente observável [13], baseados em planejamento [20] com ou sem*agendas* de execução. Um exemplo de gerenciamento baseado em agenda é o Ravenclaw [21].

III. A Arquitetura do Sistema do Adaptalker

O sistema em que o framework Adaptalker é utilizadofoi projetado para que os componentes tenham acoplamento mínimo entre si. Foi utilizado o padrão *blackboard*[22] para integração entre os componentes, de forma que eles nunca precisem se conhecer.

Outra característica desta arquitetura é o suporte a um modelo incremental, de uma forma muito similar às arquiteturas propostas em [17] e [18]. Buffers são utilizados para comunicação de eventos de atualização, compromisso e revogação de hipóteses e eventos para sinalizar fim e início de ação.Esses buffers podem trafegar hipóteses de cada um dos tipos de dados suportados.

Para preservar estas características também no gerenciador de diálogo, é necessário definir os tipos de dados de entrada e saída.

A. Entradas

Existem três tipos de ações que resultam em entradas no gerenciador de diálogo: falas, gestos e ações não comunicativas.

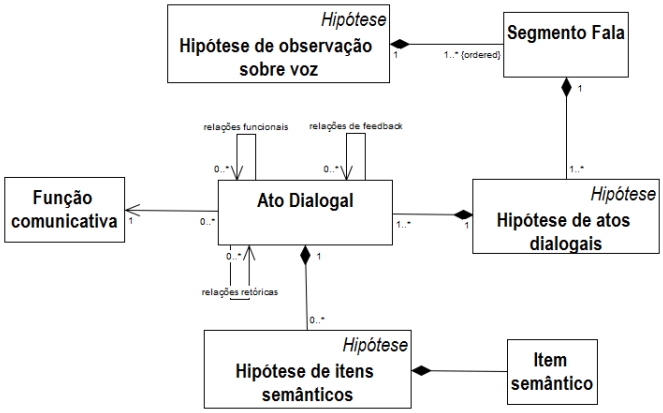

Figura 2: Modelo de entrada do gerenciador de diálogo para voz

Apenas o modelo para falas foi detalhado e implementado. Este modelo é baseado na norma ISO 24617-2 e é exibido na figura 2. Existe uma proposta para modelo de gestos feita em[1], mas cuja validade ainda não foi demonstrada. Eventos não comunicativos podem usar a mesma estrutura da figura 2, utilizando funções comunicativas distintas daquelas dos eventos comunicativos.

O conjunto de funções comunicativas suportadas deve ser o mesmo para todos os módulos que as utilizam. A função comunicativa deve especificar quais são os itens semânticos e relacionamentos opcionais e obrigatóriosassociados aos atos dialogais. O formato dos itens semânticos também deve ser o mesmo. Pode-se utilizar um modelo bastante restrito para representar os itens semânticos, como pares variáveis-valor(como em [15] ou no Segundo Desafio de Rastreamento de Estado [23]) ou utilizar uma representação mais flexível, através de alguma linguagem descritiva (como em [1]).

B. Saídas

A principal saída do gerenciador é a próxima ação a ser executada pelo sistema. Tal ação é composta de um ou mais atos dialogais. Se for utilizado um modelo nãoincremental, a próxima ação é gerada uma única vez. Se incremental, o gerenciador pode adicionar atos dialogais à saída aos poucos, gerando eventos de atualização de hipótese.

Outra saída é a lista de expectativas sobre a próxima ação do usuário, que também é composta de atos dialogais. Essa lista pode ser utilizada pelos módulos de reconhecimento de voz e entendimento de linguagem falada para refinar as hipóteses sobre o próximo turno do usuário.

IV. Gerenciamento Adaptativo de Diálogo

O modelo de gerenciamento adaptativo do Adaptalker é descrito detalhadamente em [1].O modelo é formado por duas camadas: o *filtro* e o *dispositivo adaptativo de tomada de decisão* (doravante referenciado como DATD).

O*filtro* é a camada superior, não adaptativa. A sua principal responsabilidade é decidir quais eventos devem ser descartados. Seu modelo atual é não incremental e, portanto, apenas eventos de nova ação e de fim de ação são considerados. Outra responsabilidade dele é manipular múltiplas hipóteses.Para cada hipótese com grau de confiança maior que um parâmetro c_{min}, o filtro faz uma cópia do DATD, para quem passa a hipótese a ser processada. Quando todos os DATD finalizam o processamento de suas hipóteses, é calculado um grau de confiança a posteriori c_p para cada um, multiplicando o grau de confiança na última ação do usuário pelo grau de confiança da próxima ação do sistema. A saída associada ao maior c_p é considerada e os demais DATD são descartados. Se não é possível escolherapenas uma hipótese, um procedimento de tratamento de erro é disparado.

O DATD é um dispositivo adaptativo que organiza as regras de tomada de decisão em submáquinas:

$$DATD = (M, O, \Psi_s, M_a, M_0) \quad (1)$$

Na fórmula (1), M é o conjunto de todas as submáquinas de DATD; O é o alfabeto de entrada, isto é, o conjunto de todos os atos dialogais que podem ser observados a partir de uma ação do usuário; Ψ_s é o alfabeto de saída, isto é, o conjunto de todos os atos dialogais que podem ser utilizados como saída pelo sistema; M_aé uma lista ordenada de submáquinas ativas. Cada vez que uma nova máquina é ativada, ela entra na posição inicial da lista. Quando a máquina é inativada, ela é removida da lista. Se M_a ficar vazia, não será possível a continuidade do diálogo utilizando o dispositivo; M_0 é a configuração inicial de M_a ao início de um diálogo.

Cada submáquina $\mu \in M$ é definida por uma sétupla:

$$\mu = (Q, B, T, F, X, q_0, Q_F) \quad (2)$$

em que Q é o conjunto de estados de μ;B é o conjunto de estados de *binding* - estados especiais utilizados para o vínculo de atos do usuário com as expectativas;T é o conjunto de transições de estado de μ;F é o conjunto de funções adaptativas que podem ser utilizadas em μ;X é o conjunto de variáveis que podem ser utilizadas em μ para armazenar resultados de consultas entre as transições;q_0é o estado inicial de μ, $q_0 \notin B$; Q_f é o conjunto de estados de aceitação de μ.

Cada transição pertencente a T é definida por uma óctupla

$$\gamma_{i,\pi} = (q_i, q_j, \pi, c_\mu, \alpha_a, \alpha_p, \beta, foco) \quad (3)$$

em que:

- q_i é o estado inicial da transição;
- q_j é o estado final da transição. Se $q_i \in B$, então $q_j \notin B$;
- π é a condição para que a transição seja disparada, utilizando comparadores lógicos, identificadores de variáveis $x \in X$ e literais. Se $q_i \in B$, então π também pode ser definida através de filtros de atos dialogais. A transição pode ter uma condição vazia;
- c_π é a expectativa que o gerenciador de diálogos tem de que a condição π será satisfeita. Sua definição é opcional e permitida apenas se $q_i \in B$;
- α_aé uma ação adaptativa executada antes da transição;
- α_pé uma ação adaptativa executada após a transição;
- β é a tarefa que será executada ao final da transição;
- *foco*: diz quem receberá o foco de execução das regras após a execução da tarefa β, dentre o usuário, a submáquina atual, outra submáquina ou neutro.

Uma tarefa pode ser classificada em um de quatro tipos: requisição ao usuário, informação ao usuário, chamada de submáquina e tarefa de sistema. α_ae $\alpha_p$são utilizadas para chamar funções adaptativas. Uma função adaptativa é definida em termos de parâmetros, variáveis, geradores e ações elementares, de forma similar ao mecanismo utilizado para autômatos adaptativos[24][25]e detalhado para o DATD em [1]. Podem ser utilizadas ações elementares de inserção, remoção, consulta, alteração, comparação entre duas variáveis e chamada de ação adaptativa não elementar, sendo queas três últimas podem ser definidas a partir das três primeiras [26].

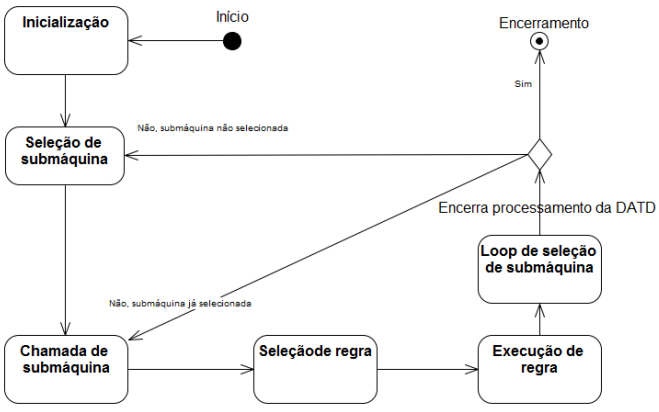

Figura 3 – Ciclo de vida de um DATD

O ciclo de vida de um DATD pode ser visto na figura 3. A inicialização é executada uma única vez, no início do diálogo.Nela, as submáquinas em M_0 são colocadas em M_a, em suas configurações iniciais. A cada turno do usuário, um DATD recebe uma lista de entrada ζ contendo atos dialogais ordenados representando uma única hipótese. Esta lista é utilizada para selecionar em qual das submáquinas ativas será a próxima execução de regra. Cada submáquina deve indicar o quão distante os atos $\varphi_i \in \zeta$estão de suas expectativas atuais. A submáquina com expectativas mais próxima é selecionada para execução.

O loop de seleção inativa submáquinas que chegaram a um estado de aceite e decide quando o dispositivo deve parar de processar e esperar o próximo turno do usuário. A inativação é importante para reduzir o espaço de procura por regras.A parada de processamento é feita em duas situações: (i) quando ζestiver vazia e não há requisições pendentes referentes ao turno anterior do sistema e (ii) quando não houver submáquinas ativas (neste caso o encerramento é definitivo). No caso da situação (i), se existirem atos dialogais de saída Φ esperando execução, o processamento é parado para processarΦ e obter a próxima ação do usuário. Se Φ está vazio, então existe um erro que causa o fim do DATD, pois nem o usuário nem o sistema estão com a inciativa – este erro acontecerá somente se houver algum erro de projeto das submáquinas.

V. O Framework do Adaptalker

Em [1], foi feita uma implementação em linguagem Java para uma aplicação de venda de pizzas por telefone, demonstrandoo funcionamento do Adaptalker, utilizando-se OWL[27] para descrição dos itens semânticos e das ontologias de memória semântica e o OWL API [28] e o *reasoner* Hermit [29] para manipulação das ontologias. Esta implementação criou seis submáquinas. Delas, destaca-se a submáquina para gerenciamento de erros, cujas regras poderiam ser reutilizadas em outras aplicações. Ela trata, por exemplo, situações em que nenhuma regra foi encontrada para algum ζ e quando não foi possível entender a ação do usuário de forma total ou parcial.

Os trabalhos atuais consistem em reorganizar ou mesmo refazer partes do código de tal forma que novas aplicações possam ser desenvolvidas facilmente utilizando o Adaptalker a partir de uma biblioteca Java, requerendo apenas aimplementação de interfaces para tratamento dos atos dialogais específicos e a produção de regras de tomada de decisão do DATD também específicos da aplicação final.

Outro trabalho importante em andamento se refere à construção dos outros módulos. O Adaptalker foi testado apenas isoladamente, com entradas produzidas manualmente por um operador. Um problema que fica mais evidente apenas com todos os módulos integrados é o tratamento dos atos dialogais e da memória semântica, pois eles não são utilizados apenas no gerenciador, mas também nos módulos de entendimento e de geração.

VI. Conclusão

O Adaptalker se mostrou adequado para o tratamento de gerenciamento de diálogo e o trabalho em andamento no framework se mostra promissor.A adaptatividade permitiu

aperfeiçoar o modelo original, como demonstrado em [1], sem remover as vantagens que um gerenciador baseado em estados finitos possui: a facilidade de desenvolvimento e de testes. É possível descrever um DATD sem ações adaptativas, da mesma forma que no dispositivo original, adicionando adaptatividade para adicionar expressividade, facilitando o projeto de regras de reparo, de iniciativa mista e de situações com dependência de contexto.

REFERÊNCIAS

[1] D. A. Alfenas, "Aplicações da tecnologia adaptativa no gerenciamento de diálogo falado em sistemas computacionais," dissertação de mestrado, Escola Politécnica da USP, 2014.

[2] D. A. Alfenas and M. R. Pereira-Barretto, "Adaptatividade em Robôs Sociáveis: uma Proposta de um Gerenciador de Diálogos," in *Memórias do WTA 2012 Sexto Workshop de Tecnologia Adaptativa*, 2012.

[3] D. A. Alfenas, M. R. Pereira-Barretto, and R. dos S. Paixão, "Sobre o uso de formalismos adaptativos no gerenciamento de diálogos em robôs sociáveis," in *MEMÓRIAS DO WTA 2013 SÉTIMO WORKSHOP DE TECNOLOGIA ADAPTATIVA*, 2013.

[4] J. José Neto, "Um Levantamento da Evolução da Adaptatividade e da Tecnologia Adaptativa," *IEEE Lat. Am. Trans.*, vol. 5, no. 7, 2007.

[5] M. F. McTear, "Spoken Dialogue Technology : Enabling the Conversational User Interface," *ACM Comput. Surv.*, vol. 34, no. 1, pp. 90–169, 2002.

[6] T. H. Bui, "Multimodal Dialogue Management - State of the art," Centre for Telematics and Information Technology, University of Twente, Enschede, Holanda, 2006.

[7] H. Sacks, E. A. Schegloff, and G. Jefferson, "A Simplest Systematics for the Organization of Turn-Taking for Conversation," *Language (Baltim).*, vol. 50, no. 4, pp. 696–735, 1974.

[8] H. Bunt, "Dimensions in Dialogue Act Annotation," in *Proceedings of the Fifth International Conference on Language Resources and Evaluation*, 2006, pp. 919–924.

[9] H. Bunt, J. Alexandersson, J. Carletta, J. Choe, A. C. Fang, K. Hasida, K. Lee, V. Petukhova, A. Popescu-belis, L. Romary, C. Soria, and D. Traum, "Towards an ISO standard for dialogue act annotation," in *Proceedings of LREC 2010, the Seventh International Conference on Language Resources and Evaluation*, 2010, pp. 2548–2555.

[10] M. Baker, T. Hansen, R. Joiner, and D. Traum, "The role of grounding in collaborative learning tasks," in *Collaborative learning: Cognitive and computational approaches*, Elsevier Science, 1999, pp. 31–63.

[11] Tomasello, "Human Cooperative Comunication," in *Origins of Human Communication*, Cambridge, MA: MIT Press, 2008, pp. 57–108.

[12] H. H. Clark and S. E. Brennan, "Grounding in communication," in *Perspectives on socially shared cognition*, L. B. Resnick, J. M. Levine, and S. D. Teasley, Eds. Washington DC: American Psychological Association, 1991, pp. 127–149.

[13] B. S. Young, M. Gasic, B. Thomson, and J. D. Williams, "POMDP-Based Statistical Spoken Dialog Systems : A Review," *Proc. IEEE*, vol. 101, no. 5, 2013.

[14] B. Dumas, D. Lalanne, and S. Oviatt, "Multimodal Interfaces : A Survey of Principles , Models and Frameworks," in *Human Machine Interaction*, D. Lalanne and J. Kohlas, Eds. Springer Berlin Heidelberg, 2009, pp. 3–26.

[15] B. Thomson, "Statistical methods for spoken dialogue management," doctoral thesis, University of Cambridge, 2009.

[16] J. D. Williams and S. Young, "Partially observable Markov decision processes for spoken dialog systems," *Comput. Speech Lang.*, vol. 21, no. 2, pp. 393–422, Apr. 2007.

[17] D. Schlangen and G. Skantze, "A General , Abstract Model of Incremental Dialogue Processing," in *Proceedings of the 12th Conference of the European Chapter of the ACL*, 2009, no. April, pp. 710–718.

[18] D. Traum, D. Devault, J. Lee, Z. Wang, and S. Marsella, "Incremental Dialogue Understanding and Feedback for Multiparty , Multimodal Conversation," in *Intelligent Virtual Agents*, Springer Berlin Heidelberg, 2012, pp. 275–288.

[19] V. V. Petukhova, "Multidimensional Dialogue Modelling," Ph.D thesis, Tilburg University, 2011.

[20] S. Kang, Y. Ko, and J. Seo, "Generating Effective Responses for a Plan-based Dialogue System in Human-robot Interaction," *INFORMATION-AN Int. Interdiscip. J.*, vol. 15, no. 2, pp. 949–960, 2012.

[21] D. Bohus and A. I. Rudnicky, "The RavenClaw dialog management framework: Architecture and systems," *Comput. Speech Lang.*, vol. 23, no. 3, pp. 332–361, Jul. 2009.

[22] D. Deugo, M. Weiss, and E. Kendall, "Reusable patterns for agent coordination," in *Coordination of Internet agents*, A. Omicini, F. Zambonelli, M. Klusch, and R. Tolksdorf, Eds. Berlin: Springer, 2001, pp. 347–368.

[23] K. Georgila and M. Stone, Eds., "15th Annual Meeting of the Special Interest Group on Discourse and Dialogue," in *Proceedings of the Conference*, 2014, no. June.

[24] J. José Neto, "Contribuiçoes à metodologia de construçao de compiladores," tese de doutorado, USP, São Paulo, Brasil, 1993.

[25] J. José Neto, "Adaptive Automata for Context-Sensitive Languages," *SIGPLAN Not.*, vol. 29, no. 9, pp. 115–124, 1994.

[26] D. A. Alfenas, D. P. Shibata, J. José Neto, and M. R. Pereira-Barretto, "Sistemas de Markov Adaptativos: Formulação e Plataforma de Desenvolvimento," in *Memórias do WTA 2012 Sexto Workshop de Tecnologia Adaptativa*, 2012.

[27] C. Bock, A. Fokoue, P. Haase, R. Hoekstra, I. Horrocks, A. Ruttenberg, U. Sattler, and M. Smith, "OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition)," *W3C Recommendation*, 2012. [Online]. Available: http://www.w3.org/TR/owl2-syntax/. [Accessed: 20-Jul-2014].

[28] M. Horridge and S. Bechhofer, "The owl api: A java api for owl ontologies," *Semant. Web*, vol. 2, pp. 11–21, 2011.

[29] I. Horrocks, B. Motik, and Z. Wang, “The HermiT OWL Reasoner,” pp. 1–6.

Daniel Assis Alfenasformou-seem Engenhariada Computação pela Escola Politécnica da Universidade de São Paulo (EPUSP)em 2004.Trabalhou, nos últimos dez anos, nas diversas áreas do desenvolvimento de sistemas, desde a definição da demanda até a implantação do sistema. Recentemente tem trabalhado como coordenador técnico no desenvolvimento de sistemas complexos que envolvem simulação, otimização e interfaces ricas. É mestrando em Engenharia de computaçãopela EPUSP, com pesquisa na aplicação da tecnologia adaptativa no gerenciamento de diálogo em linguagem naturalentre usuários e sistemas.

Carlos Eduardo Bittencourt Bogik é bacharel em Ciências da Computação pela Pontifícia Universidade Católica de São Paulo (PUC-SP) em 2003. Trabalha como programador em linguagem Java e analista de sistemas desde 2002, principalmente com sistemas Web para instituições financeiras do Brasil.

Danilo Picagli Shibata é mestre em Engenharia da Computação pela Escola Politécnica da USP (EPUSP), com título obtido em 2008 pelo estudo da aplicação de autômatos adaptativos na tradução texto-fala para textos escritos na língua portuguesa. Trabalhou com análise de segurança e confiabilidade de sistemas computacionais aplicados a sistemas críticos, desenvolvimento de software na área de segurança da informação e no desenvolvimento de simuladores de transporte de fluídos. Hoje é arquiteto de sistemas desenvolvidos em Java.

Raphael Pinheiro da Silva formou-se em Engenharia de Computação pela Escola Politécnica da Universidade de São Paulo (EPUSP) em 2004 e concluiu o curso de MBA em Inovação e Tecnologia pela Fundação Instituto de Administração (FIA) em 2013. Atua na área de desenvolvimento de software desde 2002. Atualmente é arquiteto de sistemas e líder técnico de equipe de desenvolvimento atuando principalmente com soluções na linguagem JAVA.

Marcos Ribeiro Pereira-Barretto é graduado em Engenharia Elétrica (1983), mestre em Engenharia Elétrica (1988) e doutor em Engenharia Mecânica (1993) pela Escola Politécnica da Universidade de São Paulo. Éprofessor da Escola Politécnica da USP desde 1986. Atua nas seguintes áreas de pesquisa: robôs sociáveis, computação afetiva e arquitetura de sistemas para aplicações críticas como Automação Industrial.