Asynchronous Remote Management of Agents for Subverse Project

S. P. Zambiasi, L. P. Z. B. Oberderfer and Benin M. R.

***Abstract*— The main idea of Subverse Project is focused on not only creating a virtual world for the interaction of agents. It also has the goal of being a development laboratory for computing students. Several agents can connect in this virtual world and interact with each other. However, if the agent lose connection with virtual world, then it can not control its avatar any more. A backup system (safe-mode) in the avatar may be sufficient to manage it until the connection can be restored. Thus, this paper presents a proposal for management of agents in the virtual world of Subverse Project asynchronously.**

***Keywords*— Subverse Project, Agents, Scripts asynchronous, remote management, Services Orientend Architecture.**

I. INTRODUÇÃO

A IDEIA principal do Projeto Subverse está em não ser apenas um mundo virtual para a interação de agentes, mas também de se apresentar como um laboratório para a implementação de teorias e técnicas computacionais direcionado para alunos de cursos das diversas áreas da computação. Sendo assim, não é a intenção produzir um software propriamente dito como objetivo final, mas sim apresentar aos estudantes uma plataforma em constante desenvolvimento e evolução. A premissa disso é que no meio acadêmico deve-se buscar fornecer aos estudantes oportunidades e desafios para o desenvolvimento de softwares, técnicas, algoritmos, etc. .

O projeto Subverse é composto de um mundo virtual em que diversos agentes podem se conectar remotamente e interagir entre eles, tentando sobreviver, evoluir e trabalhar de forma a alcançar seus objetivos .

O Subverse, funcionando tal como um jogo multijogador, utiliza atualmente uma arquitetura de comunicação cliente-servidor síncrona. Os clientes se conectam e enviam mensagens para um servidor responsável pelo processamento do jogo e recebem mensagens processadas para mostrar em uma interface gráfica do usuário. Ele também se utiliza de um *middleware* de comunicação para facilitar o desenvolvimento de aplicações distribuídas, deixando o desenvolvedor livre da preocupação de escrever o código para gerenciar interações entre os processos conectados remotamente , .

Entretanto, ao se utilizar de uma forma de comunicação síncrona, podem haver atrasos de comunicação (*lags*) ou a perda da conexão, principalmente quando houverem muitas conexões ocorrendo ao mesmo tempo no servidor ou houver algum problema de comunicação na rede de comunicação em si. Concomitantemente, o agente conectado deixa de controlar seu avatar devidamente, deixando-o indefeso às atividades do mundo virtual ou à ataques de agentes inimigos.

Uma forma que se mostra atraente para resolver tal problema é a implementação de uma forma de *safe-mode* no avatar (elemento no mundo virtual controlado remotamente pelo agente). Este mecanismo de *safe-mode* caracteriza-se por uma implementação embutida no próprio avatar e que pode ser implementada ou configurada pelos usuários.

Partindo dessa ideia, duas formas diferentes de controle do avatar pelo agente podem ser verificadas: (i) o algoritmo *safe-mode* entraria em execução depois de um *timeout* de comunicação entre o avatar e seu agente e (ii) toda a comunicação entre agente e avatar poderia ser assíncrona.

Um outro problema que ocorre muitas vezes são os *firewalls* fechados de empresas ou mesmo de Instituições de Ensino, não permitindo que hajam conexões para diferentes portas, que não a porta 80. Uma alternativa a isso, e em adendo ao que já foi exposto, poder-se-ia utilizar para a comunicação entre agente e avatar uma forma padronizada, atual e assíncrona, tal como a estrutura fornecida pela tecnologia de serviços web, no estilo da Arquitetura Orientada a Serviços (SOA padrão OASIS) .

Dessa forma, este artigo apresenta uma proposta de gerenciamento dos agentes no mundo virtual do Projeto Subverse de forma assíncrona no padrão SOA e com gerenciamento e controle do avatar pelo agente via um mecanismo *safe-mode*.

Este artigo está organizado da seguinte forma. O capítulo I. introduz e contextualiza o problema e o trabalho de pesquisa. O capítulo II. apresenta uma revisão básica do assunto abordado. No capítulo III. é apresentada a proposta propriamente dita e no capítulo IV. são apresentadas as considerações finais do trabalho e os próximos passos.

II. BREVE REVISÃO DA ABORDAGEM

Uma vez apresentada a problemática básica e os objetivos da proposta, toma-se como próxima etapa a apresentação dos elementos teóricos e tecnológicos envolvidos.

A. Projeto Subverse

O Projeto Subverse caracteriza-se por se apresentar como um ambiente de aprendizado para alunos de cursos de computação. Não é a intenção que deste seja gerado um produto implementado ou software como objetivo final, por exemplo um jogo de computador online multijogador. Por

S. P. Zambiasi, Universidade Federal de Santa Catarina (UFSC), Florianópolis, Santa Catarina, Brasil, saulopz@gmail.com

L. P. Z. B. Oberderfer, Instituto Federal de Santa Catarina (IFSC), Chapecó, Santa Catarina, Brasil, docepolly@gmail.com

M. R. Benin, Pontifícia Universidade Católica do Paraná (PUCPR), Curitiba, Brasil, mbenin@gmail.com

meio do Projeto Subverse deve ser possível que os alunos possam implementar comportamentos nos agentes, que são diretamente aplicados aos seus avatares – elementos do mundo virtual que representam os agentes conectados. Também deve ser possível que os alunos possam criar novos ambientes de configuração, *middlewares, plugins*, softwares de visualização gráfica do que ocorre no mundo virtual, etc. Por meio dessa plataforma os alunos podem trabalhar, na prática, diversas teorias apresentadas em disciplinas de computação, tais como Inteligência Artificial, Algoritmos, desenvolvimento de Jogos de Computador, Sistemas Distribuídos e outras .

O Subverse possui uma estrutura de arquitetura básica (Fig. 1) composta de dois lados, o servidor com um banco de dados, um ambiente de configuração via web, um mundo virtual e avatares que interagem nesse mundo virtual, controlado pelos agentes (do lado do cliente).

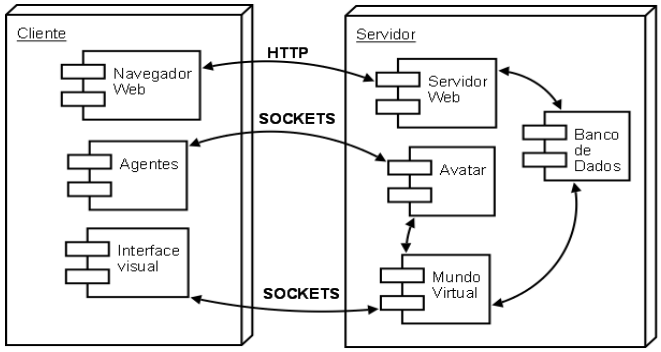

Figura 1. Visão geral da proposta.

Do lado do cliente é necessário que o usuário possua um navegador web instalado em seu dispositivo computacional. Este serve para que o usuário possa acessar o ambiente de configuração dos seus agentes. Além disso, o usuário pode instalar uma interface visual gráfica para que ele possa visualizar o que ocorre no servidor. Por fim, do lado do cliente é efetuada a implementação dos agentes, que controlam seus avatares no mundo virtual do Subverse .

Propriamente, o Subverse é composto de partes modulares de um mundo virtual para a implementação de agentes. Estes são baseados no paradigma de jogos multijogador que trabalham em rede. Essa estrutura apresenta-se aos alunos como um desafio de implementações em constante evolução. O Projeto Subverse também sugere a utilização de tecnologias *opensource,* atuais e disponíveis de forma gratuita .

Esta estrutura distribuída para múltiplas conexões de agentes remotos para interação no mundo virtual do Subverse pode ser vista, comparativamente, na forma de um *Massive Multiplayer Online Game* (MMOG). Porém, diferentemente de um jogo de computador, ou console, não são pessoas que se conectam ao servidor para jogar, mas sim são agentes pré-programados que se conectam ao servidor com o propósito de gerenciarem/controlarem seus avatares . No Subverse, os usuários podem implementar e executar seus agentes. Cada agente possui um avatar representante no mundo virtual que precisa estar cadastrado no servidor via interface web. Estes se conectam ao mundo virtual e iniciam seu ciclo de vida. O usuário pode então visualizar o que está acontecendo por meio de uma interface visual gráfica.

B. Agentes

Como o Subverse apresenta agentes como elemento base para gerenciar os avatares no mundo virtual, uma explanação sobre essa tecnologia torna-se necessária neste ponto do artigo.

Segundo , agentes podem perceber o meio ambiente em que se encontram através sensores e podem responder com ações de forma a atingir os seus objetivos. Não obstante, um agente precisa se comunicar com outros agentes e/ou outros sistemas para poderem alcançar seus objetivos. Dessa forma, este passa a participar inevitavelmente de um sistema multiagente.

Conforme "um agente é tudo aquilo que pode perceber o ambiente em que se encontra, através de sensores, e agir sobre este ambiente por meio de atuadores". Assim como um agente humano dispõe de sensores (olhos, ouvidos, tato, etc.) e atuadores (mãos, pernas, etc.), um agente robô, por exemplo, pode ser provido de sensores, tais como câmeras, infravermelhos e possuir atuadores, como rodas, braços mecânicos e diversos tipos de motores.

Em complementação, um agente pode ser visto como um elemento físico ou virtual com capacidade de atuar em um ambiente. Este pode se comunicar diretamente com outros agentes, possuir objetivos individuais de satisfação e sobrevivência, recursos próprios, ser capaz de perceber o ambiente, possuir habilidades, oferecer serviços, ser capaz de se reproduzir e possuir comportamentos que visam satisfazer seus objetivos .

Para , o princípio básico da utilização de agentes é que eles devem "saber das coisas" e o objetivo da Inteligência Artificial é projetar agentes que façam um bom trabalho com ações satisfatórias em seu ambiente.

Já para , o agente é definido por um sistema de computador situado em algum ambiente e com capacidade de ações autônomas neste ambiente. Este ambiente pode ser tanto real (agentes de hardware) como virtual (agentes de software). Ainda, o agente deve ser capaz de perceber o meio em que se encontra e responder satisfatoriamente as mudanças que ocorrem no mesmo para poder cumprir com seus objetivos com certa autonomia flexível. A principal distinção destes em relação a agentes inteligentes é a capacidade de tomar iniciativas, a fim de satisfazer seus objetivos e pela aptidão de interagir com outros agentes e, possivelmente, com pessoas.

Outras definições de agentes concebem um refinamento da tecnologia em um sentido mais avançado, tais como a definição de um agente racional. Este busca realizar uma ação que maximiza seu desempenho para cada possível sequência de percepção. Tal especificidade é chamada de mapeamento ideal e toma como base para suas deliberações evidências fornecidas pela sequência de percepção e pelas informações já existentes em sua base de conhecimento .

Para , na computação, o termo agente abrange uma ampla gama de comportamentos e funcionalidades. Para ele, um agente ativo computacional é uma entidade que: tem uma identidade persistente, necessária para realizar transações, manipular exceções e para construir uma história de interações para definir a confiança com os outros agentes da qual este interage; pode perceber e responder ativamente às suas atividades no seu ambiente e pode se comunicar com outros agentes, ou pessoas. Além disso, os níveis de autonomia para estes agentes variam conforme as restrições do ambiente em que se encontram.

A autonomia desses agentes depende da quantidade de sensores, atuadores e informações que eles possuem, além disso, grau de autonomia está relacionado à capacidade de decidir por si só como relacionar as informações dos sensores com seus atuadores para atingir seus objetivos e satisfazer suas motivações. Os agentes autônomos trabalham em ambientes dinâmicos e imprevisíveis, interpretando as informações obtidas pelos sensores e atuando, como resposta, no ambiente .

De fato, as abordagens da utilização das tecnologias de Agentes e Sistemas multiagentes (MAS) já são bastante conhecidas e vêm sendo muito utilizadas para engenharias complexas, flexíveis e aplicações orientadas a serviços .

C. Arquitetura Orientada à Serviços

Conforme definido nos objetivos deste artigo, a Arquitetura Orientada a Serviços (*SOA – Service Oriented Architecture*) é sugerida para ser utilizada como base para a comunicação assíncrona entre o agente e seu avatar no mundo virtual do Subverse. Em verdade, o SOA foi escolhido como Estilo Arquitetural para que sirva como norteador dos princípios de funcionamento de controle do avatar e interoperabilidade. Dessa forma, se torna necessária uma explanação sobre o assunto ao leitor, situando-o de certas especificidades e apontando alguns importantes conceitos. Também, não convém extrapolar, ou estender exaustivamente o assunto referente à SOA, mas sim colocar o leitor a par dos termos utilizados na proposta.

A Arquitetura Orientada a Serviço (SOA), conforme "é um paradigma para organização e utilização de competências distribuídas que estão sob o controle de diferentes domínios proprietários". Ou seja, as pessoas e organizações criam competências para resolver problemas específicos conforme suas necessidades. Estas são modeladas através de um conjunto de componentes que compõe a arquitetura e podem ser invocados por meio da descrição de suas interfaces, que podem ser publicadas e descobertas. .

Segundo , o ponto central está no conceito de serviço e consumidores de serviços. Sendo assim, a tecnologia de serviços web é só uma maneira de aplicar a teoria na prática. Por meio da utilização de SOA, há uma negociação entre fornecedor e consumidor por meio de uma interface comum.

A tecnologia da Arquitetura Orientada a Serviços surgiu como forma de reduzir o tempo, custos e recursos gastos na tentativa de uma integração rápida e flexível dos sistemas de Tecnologia da Informação e Comunicação envolvidos nas organizações. A utilização de SOA enfoca no aumento da agilidade na implementação de novos produtos e processos e na composição de novos processos por meio da orquestração de componentes já prontos. Além disso, abrange a possibilidade de reuso de sistemas e aplicações legadas. Esta tem como base componentes baseados em serviços, para que se possa atingir essa integração de forma rápida e flexível .

Na arquitetura SOA, as aplicações de negócios do cotidiano são divididas em funções e processos de negócios individuais, compartilhados e reutilizáveis. Estes, chamados de serviços, são a composição básica da arquitetura SOA e podem fazer parte da composição de outros serviços pela integração/orquestração destes, independente das aplicações e plataformas computacionais para a execução .

Para tornar estes serviços disponíveis, e viabilizados para o uso por terceiros, deve ser disponibilizada uma descrição deste serviço contendo as informações necessárias para a interação com ele e descrevendo suas entradas, saídas e semânticas associadas à eles .

As interfaces de todas as funções devem estar bem definidas na forma de serviços independentes com interfaces de invocação bem definidas. Estas podem ser implementadas com várias tecnologias. Contudo, a principal tecnologia empregada hoje por diversas empresas por possuir um meio padronizado de implementação e a um custo razoável é a tecnologia de serviços web .

Um dos principais objetivos da utilização de serviços web é a implementação de SOAs, por ser a tecnologia que melhor atende seus requisitos. Estes requisitos, segundo , são:

- Baixo acoplamento: Os serviços da arquitetura não devem ter uma dependência forte entre si.
- Independência de implementação: Não se deve depender de características específicas de linguagens de programação ou ambientes de execução.
- Configuração flexível: Os diferentes serviços devem poder ser ligados entre si de forma dinâmica e configurável.
- Tempo de vida longo: Os serviços devem existir por tempo suficiente para que sejam descobertos e utilizados até se obter confiança em seu comportamento.
- Granularidade: As funcionalidades de um sistema devem ser divididas em vários serviços.
- Distribuição: Os serviços devem ficar distribuídos, para aproveitar melhor os recursos computacionais.

A arquitetura SOA combina a habilidade de invocar objetos remotos e funções com ferramentas para a descoberta dinâmica de serviços, com ênfase na interoperabilidade . Entretanto, a arquitetura SOA não é, por si só, uma solução para problemas para um dado domínio, e sim um paradigma de organização e entrega que permite obter mais valor das competências localmente disponíveis e daquelas controladas por terceiros .

Um serviço, por sua vez, é um componente de software caracterizado como uma implementação bem definida de uma

funcionalidade de negócios e com uma interface que pode ser publicada e descoberta por consumidores de serviços. Estes componentes podem ser agrupados para a criação de diferentes aplicações e processos de negócios utilizando-se de um modelo de comunicação baseado na troca de mensagens com baixo acoplamento . Estes buscam a interoperabilidade através da interação entre um computador e uma rede e se comunicam entre eles e outros sistemas por meio de mensagens baseadas no protocolo HTTP (*Hipertext Trasnfer Protocol*). Os três elementos principais dos serviços web são: (i) o protocolo de comunicação, (ii) a descrições dos serviços e (iii) a descoberta de serviços. .

Segundo , os serviços Web trouxeram um novo paradigma suportando sistemas distribuídos fracamente acoplados com a descoberta e execução de serviços.

A base para esses três elementos é a linguagem XML (*Extensible Modeling Language*). O protocolo de comunicação entre os serviços Web e seus clientes é o SOAP (*Simple Object Access Protocol*), o WSDL (*Web Services Definition Language*) é a linguagem que provê um mecanismo para descrever formalmente os serviços Web, utilizando-se de *XML Schema* para especificar a estrutura e os tipos dos dados que estão contidos nas mensagens e, por fim, o UDDI (*Universal Descripition, Discovery and Integration*) é o diretório de registro de serviços, acessados por meio de mensagens SOAP .

É importante observar que inicialmente, SOAP significava Protocolo Simples para Acesso a Objetos, mas ele passou a ser apenas um nome em versões mais recentes, pois ele foi considerado um acrônimo incorreto se comparado com a funcionalidade atual do protocolo .

A linguagem XML é uma linguagem de marcação de propósito geral que pode ser utilizada para a criação de linguagens de marcação com propósitos específicos, chamadas de dialetos. Com a linguagem XML é possível representar diferentes tipos de dados, além de fornecer recursos para combinar textos e informações extras sobre estes. Isso significa que trechos do texto podem conter informações de como este trecho específico do texto deve ser representado. A especificação 1.0 da XML apresenta uma sintaxe que define uma forma de se criar linguagens específicas, tal como a DTD (*Document Type Definition*), ou Definição de Tipo de Documento. Porém, com a utilização do *XML Schema*, o DTD se tornou obsoleto .

A linguagem *XML Schema*, por sua vez, se utiliza da linguagem XML para a criação de esquemas de validação para XML. Um documento *XML Schema* é constituído por declarações de elementos, atributos e tipos. A cada elemento ou atributo é associado um tipo, o que pode ser declarado juntamente com o elemento, como no caso dos elementos mensagem e destinatário ou referenciando-se um tipo declarado separadamente através de um atributo *type.* Este tipo pode ser declarado no mesmo documento ou importado de outro documento *XML Schema*, podendo ser tipos simples ou complexos .

D. Implementação Embutida

Segundo , a maioria dos jogos se utilizam do auxílio de alguma linguagem de *script* embutida no software. Essas linguagens normalmente são interpretadas, com tipagem dinâmica e gerenciamento automático de memória. A construção de estruturas de informações são bastante dinâmicas, fornecendo aos usuários uma forma interessante de ampliar as possibilidades do software em tempo de execução e sem a necessidade de uma nova compilação do código do software implementado.

O funcionamento dessa ideia se dá por meio da implementação de um *script* em um programa hospedeiro, desenvolvido em uma linguagem de programação não necessariamente igual ao programa em si (por exemplo, o *script* em linguagem LUA e o programa em C++). Uma outra característica, que somada com a anterior torna a *scriptagem* uma excelente ferramenta de desenvolvimento, é que essas linguagens também não devem poder acessar execuções não autorizadas no programa hospedeiro .

Várias vantagens podem ser identificadas pela utilização de uma linguagem de *script* acoplada em um software (normalmente utilizadas em jogos de computador). O *script* pode ser utilizado para definir configurações, objetos e seus comportamentos, gerenciar algoritmos de inteligencia artificial e tratar os eventos de entrada .

Em , a linguagem Lua é apresentada como uma linguagem de *script* bastante atual e como uma das mais utilizadas no desenvolvimento de jogos. Isso devido "ao seu pequeno tamanho, bom desempenho, portabilidade e facilidade de integração". Essa linguagem é extensível e projetada para oferecer metamecanismos que facilitam a adequação da linguagem às necessidades da aplicação.

III. PROPOSTA DE GERENCIAMENTO ASSÍNCRONO

A proposta deste artigo está embasada em alguns pontos básicos: (i) os agentes podem, por algum motivo, perder a conexão de rede, não mais podendo ter o controle sobre seus avatares para sobreviverem no ambiente virtual de forma segura. Assim, o avatar pode ficar sem ação, podendo não cumprir devidamente com os objetivos do agente ou sendo atacado por algum inimigo ou ficando sujeito às ameaças do ambiente virtual onde ele se encontra; (ii) A tecnologia da Arquitetura Orientada à Serviços (SOA) oferece uma forma de conexão assíncrona que, além de ser uma forma alternativa de conexão ao avatar do agente, é baseada em padrões abertos e interoperáveis para se trabalhar em um estilo de desenvolvimento modular e distribuído baseado em *software-as-a-service*; e (iii) A linguagem Lua, já conhecida e utilizada por diversos desenvolvedores para a criação de código embutido em um programa (*script*), é apresentada aqui como sugestão de criação dos *scripts* de comportamentos do tipo *safe-mode*, que podem manter o avatar em operação enquanto a conexão com o agente não é restabelecida.

Baseando-se nos pontos apresentados acima, é apresentada na Fig. 2 uma visão geral da proposta desse artigo.

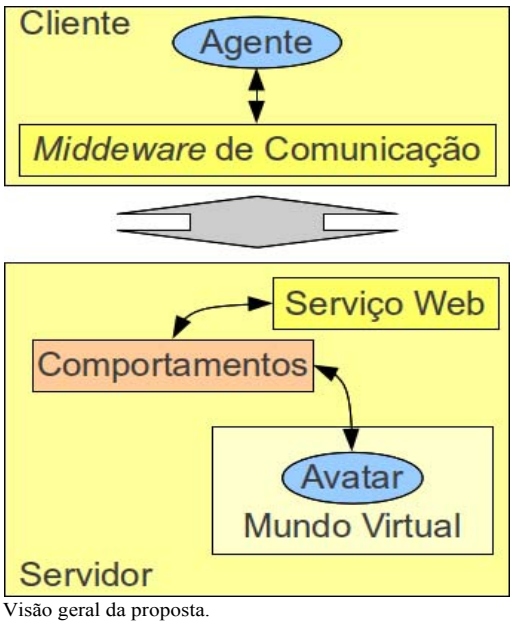

Figura 2. Visão geral da proposta.

Do lado do cliente, continua existindo um agente que se utiliza de um *middleware* de comunicação com o avatar e o servidor, tal como apresentado em . Este é provido de métodos para gerenciar o avatar. Entretanto, agora do lado do servidor, a forma de conexão se dá pelo acesso à serviços web. Por meio de um conjunto de operações que o *middleware* fornece, ações podem ser atribuídas para serem executadas no avatar do agente e requisições de informações e contexto podem ser retornadas para que o agente esteja a par do que se passa com seu avatar. Também é importante agora considerar que a conexão não é mais síncrona, e a atividade do avatar necessita ser melhor planejada.

Em tempo, como não há mais uma conexão direta para o controle do avatar via agente, nas configurações do avatar do lado do servidor é acrescentado um novo módulo, aqui chamado simplesmente de Comportamentos. Estes se caracterizam pelo somatório de *scripts* escritos em linguagem de programação Lua, cadastrados pelo usuário no avatar via ambiente web de configuração e que podem ser acessados/utilizados pelo agente remotamente.

Com este recurso, várias ações mais complexas podem ser tomadas pelo avatar, comandado pelo agente. Porém, devido ao fato da conexão não mais ser síncrona, é necessário que a implementação do lado do agente seja focada agora para o planejamento, isto é, o usuário deve programar uma sequência de ações que podem ser executadas pelo avatar. Um exemplo de autonomia de execução que o avatar pode possuir, é a forma utilizada pelo Rover em Marte. Para que as pessoas na terra possam controlar este dispositivo robótico foi necessário que ele tivesse certa autonomia. Isso pois há um *delay* de comunicação entre o planeta Terra e o planeta Marte, não sendo possível controlar o sistema em tempo real. Em caso de pane, obstáculos e outros problemas, o robô em Marte necessita encontrar a solução sem o auxílio obrigatório das pessoas na Terra .

Não obstante, há de se considerar que o controle do avatar via agente, pode se dar pela soma das duas formas de comunicação. Se alguma dessas falhar, outra (*backup*) é estabelecida. Ou mesmo, se o avatar ficar sem o controle do agente, o módulo de comportamentos autônomos, desenvolvidos pelo usuário via *scripts* em linguagem de programação Lua, assume o controle do avatar.

IV. CONSIDERAÇÕES FINAIS

Este artigo apresentou uma proposta de controle do avatar pelo agente no mundo virtual do Projeto Subverse via comunicação Assíncrona. Para tal, sugere-se a utilização da Arquitetura Orientada a Serviços com serviços web, adaptando a comunicação à um protocolo padrão e interoperável. Em tempo, a proposta sugere também a utilização de *scripts* embutidos no avatar, no servidor do Subverse. Estes implementados em linguagem de programação Lua, que se apresenta como uma ferramenta para fornecer certa autonomia ao avatar, quando este não se encontra mais em comunicação com seu agente (controlador).

A proposta foi apresentada no nível de pesquisa dos assuntos relacionados, em nível de visão geral e de forma conceitual. Um protótipo inicial baseado na proposta já foi implementado e testado. Contudo, este se caracterizou de forma muito simples e não sendo aplicado ao próprio Subverse. Dessa forma, uma nova versão, sendo agora implementada em cima da estrutura atual do Projeto Subverse encontra-se em seu período de desenvolvimento e os resultados da execução da implementação devem ser avaliados.

Em tempo, além de apresentar neste artigo novas linhas de pesquisa a serem adicionadas ao Projeto Subverse (planejamento, autonomia do avatar remotamente, linguagens de *scripts*, etc), é aqui sugerida uma forma de comunicação via SOA, padronizada, permitindo que outras instâncias de agentes possam se comunicar de maneira mais interoperável.

Contudo, a implementação ainda não foi testada e novos desafios ainda devem ser encontrados e avaliados, incluindo ao que se refere a limitações do próprio servidor do Subverse, da linguagem de *script* e da forma de comunicação. Além de ser necessário avaliar os impactos na atual metodologia de execução e comunicação dos agentes com seus avatares e propriamente do funcionamento do Sistema do Projeto Subverse como um todo.

Neste sentido, este trabalho deve ser visto como uma contribuição para o Projeto Subverse, em vários níveis. Há vários aspectos que devem ser mais bem estudados e implementados. Como próximos passos, o trabalho direciona-se ao término do protótipo do módulo a ser acrescentado ao conjunto de módulos do sistema do Projeto Subverse para testes e avaliação. Também, devem ser implementados os agentes que serão conectados de forma assíncrona ao servidor, controlando seus avatares, agora com certa autonomia.

REFERÊNCIAS

[1] Agrawal, R., J.Bayardo Jr., R., Gruhl, D., Papadimitriou, S. Vinci: a service-oriented architecture for rapid development of Web applications. Computer Networks, 39(5):523–539. <http://www.sciencedirect.com/science/article/B6VRG-45KNFVW-3/2/>. Accessed in Nov/2008. 2005.

[2] Booth, D.; Haas, H.; Mccabe, F.; Newcomer, E.; et al. Web Services Architecture. <http://www.w3.org/TR/ws-arch/> Accessed in Fev/2009. 2004.

[3] Bray, T.; Paoli, J.; Sperberg-McQueen, C.M.; at al. Extensible Marckup Language (XML). W3C Recomendation. <http://www.w3.org/TR/xml> accessed in Mar/2009. 2008.

[4] Celes, W. and de Figueiredo, L.H. and Ierusalimschy, R.. A Linguagem Lua e suas Aplicações em Jogos. Rio de Janeiro, 2004. Disponível em: <http://www.tecgraf.puc-rio.br/~lhf/ftp/doc/wjogos04.pdf> Acessado em Ago/2011.

[5] Dickey, C.G, Zappala, D. and Lo LO, V., 2004. A Distributed Architecture for massively multiplayer online games. ACM NetGames Workshop.

[6] Dudley, C.; Rieu, L.; Smithson, M.; Verma, T.; et al.. WebSphere Service Registry and Repository Handbook. First Edition ed. IBM RedBooks: IBM. 2007.

[7] Estefan, J. A.; Laskey, K.; McCabe, F.; Thornton, D. Reference Architecture for Services Oriented Architecture Version 1.0. OASIS. <http://docs.oasis-open.org/ soa-rm/soa-ra/v1.0/soa-ra-pr-01.pdf>, Accessed in Dec/2008. 2008.

[8] Ferber, J., Multi-agent systems: an introduction to distributed artificial intelligence. Addison-wesley, United Kingdom, London, 1998.

[9] Gesser, Carlos E. Uma abordagem para a integração dinâmica de serviços web em portais. Eng. Elétrica - UFSC. Dissertação. 2006.

[10] Haas, H; e Brown, A.; 2004. Web Services Glossary. <http://www.w3.org/TR/ws-gloss/>, acessado em Dez/2008.

[11] Huhns, Michael N. Singh, M.P., Personal assistants, IEEE Internet Computing, Vol. 2, Issue 5, pp. 90-92. Sep./Oct. 1998.

[12] Huhns, M.N.; Singh, M.P.; Burstein, M.; at al. Research directions for service-oriented multiagent systems. IEEE Internet Computing. vol 9, num 6, pg 65-70. 2005.

[13] MacKenzie, C.; Laskey, K.; McCabe, F. at all. Reference Model for Service Oriented Architecture 1.0. OASIS Standard, 12 October 2006. <http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf> acessado em Fev/2009.

[14] O'Brien, L.; Bass, L.; Merson, P. Quality Attributes and Service-Oriented Architectures. Technical Note - Software Engineering Institute, Carnegie Mellon University. 2005.

[15] Paurobally, S.; Jennings, N.R. Developing Agent Web Service Agreements. Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI'05). 2005.

[16] Piazza, André P. Uma abordagem para interoperabilidade entre plataformas heterogêneas de serviços web para redes colaborativas de organizações. Eng. Elétrica - UFSC. Dissertação. 2007.

[17] Russel, S., Norvig, P., Inteligência Artificial, 2aEd, Tradução da segunda edição. Rio de Janeiro: Elsevier, 2004.

[18] Singh, M. P. e Huhns, M. N. Service-Oriented Computing: Semantics, Processes, Agents. John Wiley & Sons, New York, NY, EUA, 1a. Edição. 2005.

[19] Stojanovic, Z.; Dahanayake, A.; Service-oriented software system engineering: challenges and pratices. Idea Group Inc. Hershely, PA. 2005.

[20] Thompson, H. S., Beech, D., Maloney, M., e Mendelsohn, N.; 2004. XML Schema (2nd Edition). <http://www.w3.org/TR/xmlschema-1/> acessado em Abril/2006. 2004.

[21] Wang, H., Huang, J. Z., Qu, Y., e Xie, J.Web services: problems and future directions. Journal of Web Semantics, 1(3):241–320. 2005.

[22] Wooldridge, M. An introduction to multi-agent systems. England: John Wiley and Sons. 2001.

[23] Volpe, R. Rover functional autonomy development for the mars mobile science laboratory. Proc. 2003 IEEE Aerospace Conf, pg.243-652. 2003.

[24] Zambiasi, S.P.; Benin, M.R.; Silva, F.C. Projeto Subverse: Um Mundo Virtual Baseado em Jogos Multijogadores como Ferramenta de Ensino Multidisciplinar em Cursos de Tecnologia da Informação. I Simpósio Santa Catarina Games (SCGames2009). Florianópolis, Brasil, 2009.

[25] Zambiasi, S.P.; Silva, F.C. Uma proposta de middleware de comunicação para jogos multijogador online com enfoque no Projeto Subverse. IX Simpósio Brasileiro de Jogos e Entretenimento Digital - SBGames2010. Florianópolis - SC. 2010.

Saulo Popov Zambiasi possui graduação em Ciência da Computação pela Universidade do Oeste de Santa Catarina - Campus de Chapecó (1998), especialização em Ciência da Computação pela Universidade Federal de Santa Catarina (2000) e mestrado em Ciências da Computação pela Universidade Federal de Santa Catarina (2002). Atualmente é Pesquisador – Estudante de doutorado do Programa de Pós-Graduação em Engenharia de Automação e Sistemas da Universidade Federal de Santa Catarina. É professor da Universidade do Sul de Santa Catarina (UNISUL), das Faculdades Barddal e Revisor de periódico da Revista IEEE América Latina. Tem experiência na área de Ciência da Computação, com ênfase em Computação Aplicada, atuando principalmente nos seguintes temas: Inteligência Artificial Distribuída, Sistemas Multiagentes, Agentes Inteligentes, Computação Gráfica, Jogos de Computador e Automação Residencial. É membro integrante do Projeto Subverse.

Lara Popov Zambiasi Bazzi Oberderfer é formada em Ciência da Computação pela Universidade Comunitária da Região de Chapecó (UNOCHAPECO). Possui especialização em Gestão Estratégica em Tecnologia da Informação pela Faculdade Exponencial. Trabalha atualmente como professora do Instituto Federal de Santa Catarina (IFSC) Campus de Chapecó. É membro integrante do Projeto Subverse. Tem experiência nas áreas de desenvolvimento de sistemas para internet e Inteligência Artificial.

Max Ricardo Benin possui graduação em Sistemas de Informação pela Faculdades Barddal em 2007. Pós-graduado em Desenvolvimento de Jogos Eletrônicos pela Pontifícia Universidade Católica do Paraná em 2011. Atualmente é professor de um curso livre em desenvolvimento de aplicações WEB, Analista Sênior em empresa privada e CTO e fundador de uma *startup* com foco em Desenvolvimento de Jogos Eletrônicos. Possui experiência na área de Desenvolvimento de Aplicações WEB com ênfase em semântica, padrões e WEB 2.0 e também possui experiência em Desenvolvimento de Jogos Eletrônicos multiplataforma com ênfase em Arquitetura e Padrões de Sistemas, Design e Prototipagem. É membro integrante do Projeto Subverse.