Integração dos simuladores de voo da aeronave A-29 usando a arquitetura High Level Architecture

Daniel Lélis Baggio e Luisa Amaral de Almeida
Centro de Computação da Aeronáutica de São José dos Campos - Praça Marechal Eduardo Gomes, nº 50 - Vila das Acácias, São José dos Campos - SP

***Resumo* — O avanço tecnológico na área de simulação permitiu que simuladores de voo pudessem operar em conjunto, aperfeiçoando o treinamento de voos em pacote. Nesse sentido, a High Level Architecture (HLA) foi definida como a arquitetura para prover interoperabilidade entre os simuladores do Ministério da Defesa. Como consequência, surgiu o desafio de adaptar os simuladores existentes na Força Aérea Brasileira aos padrões definidos na HLA. Assim, o presente trabalho mostra os desafios e soluções encontrados no desenvolvimento da camada de integração dos simuladores do A-29 via HLA. Como resultado, conseguiu-se prover interoperabilidade não apenas entre os simuladores do A-29, mas também com ferramentas COTS (commercial off-the-shelf) de forças geradas por computador, proporcionando o treinamento tático em conjunto. A solução ainda precisa de melhorias, principalmente para evitar erros de interpolação nas posições das entidades da simulação, além de uma avaliação operacional a fim de verificar sua efetividade no treinamento.**

***Palavras-Chave* — Simulação de cenários de Comando e Controle, High Level Architecture, Interoperabilidade.**

I. INTRODUÇÃO

A Força Aérea Brasileira conta com mais de 14 simuladores de voo, os quais são responsáveis por grande parte do treinamento operacional das aeronaves que representam. A simulação aplicada ao treinamento na Força Aérea Brasileira (FAB) pode substituir tempo de treinamento na aeronave e, consecutivamente, reduzir gastos. Nesse sentido, um conceito fundamental inerente à simulação de voo é o *Transfer of Training (TOT)*, que é a capacidade do treinamento aprendido em uma situação anterior ser transferido para uma nova situação [1].

Para medir essa transferência, é comum utilizar o índice *Transfer Effectivenes Ratio* (TER), o qual é calculado por meio da equação 1, em que A é o tempo de treinamento na aeronave real sem a utilização do simulador e A_s o tempo de treinamento na aeronave real quando utilizado um simulador por S unidades de tempo [2].

$$TER = \frac{A - A_s}{S} \tag{1}$$

A análise do TER não é suficiente para definir se um simulador é economicamente eficaz ou não, uma vez que é importante considerar o custo relativo entre treinamento real e simulado. Por sua vez, o índice *Cost Effectiveness Ratio* (CER) considera também a relação de custo, sendo definido pela equação 2 [2]. Simuladores com CER maior do que 1 são economicamente viáveis, ou seja, proporcionam economia de recursos.

$$CER = \frac{Training\ Effectiveness\ Ratio(TER)}{Training\ Cost\ Ratio(TCR)} \tag{2}$$

em que $TCR = \frac{Custo\ da\ Hora\ de\ Voo\ no\ Simulador}{Custo\ da\ Hora\ de\ Voo\ na\ \ Aeronave}$.

Um estudo realizado com os simuladores do A-29 (Super Tucano) do COMAER, no período de 2009 a 2012, mostrou que o CER médio desses simuladores é de 2,66 [3], ressaltando a economia de recursos trazida pelo treinamento simulado.

Naquela época e até o presente momento, os simuladores do A-29 do COMAER foram usados principalmente para treinamento de operações de pouso e decolagem, voo IFR e VFR, procedimentos de emergência e, em alguns casos, emprego de armamento.

Contudo, as novas tecnologias possibilitam que a simulação amplie sua abrangência ao treinamento tático de missão. Nesse sentido, a interoperabilidade entre simuladores tem se mostrado como uma solução para interligar diferentes sistemas de simulação a fim de proporcionar o treinamento operacional conjunto. Esse treinamento poderia reduzir a demanda excessiva de voos em pacote das aeronaves A-29 ao longo dos Programas de Especialização Operacional (PESOP) da Aviação de Caça e Programa de Elevação Operacional (PEVOP). Consequentemente, haveria redução dos recursos empregados pelo COMAER em voos de missão composta e aumento do CER dos simuladores.

O Centro de Computação da Aeronáutica de São José dos Campos (CCA-SJ) vem realizando pesquisas para iniciar o treinamento simulado conjunto na FAB, em especial nos simuladores do A-29. Uma das bases para o início do trabalho foi a definição da arquitetura utilizada na interligação dos simuladores. A Portaria Normativa Nº1.814/MD, de 13 de junho de 2013, determinou a *High Level Architecture* (HLA) [4] como arquitetura para prover a interoperabilidade entre os simuladores das Forças Armadas.

Em conformidade com a Portaria Normativa, a compatibilidade com HLA está sendo especificada nos requisitos dos simuladores a serem adquiridos pela FAB. Contudo, os simuladores adquiridos anteriormente (como os

simuladores do A-29) não apresentam essa característica. Dessa forma, para que o treinamento integrado também os contemplasse, foi necessário o desenvolvimento de uma camada para converter os dados disponíveis nos simuladores em dados compatíveis com HLA e vice-versa. Assim, esse trabalho apresenta os desafios e soluções encontradas no desenvolvimento da *bridge* de ligação entre os antigos simuladores do COMAER e a arquitetura HLA.

II. HLA

Em meados dos anos 90, a HLA foi desenvolvida pelo *Defense Modeling and Simulation Office* (DMSO) dos Estados Unidos, com o objetivo de prover um *framework* técnico comum para toda a comunidade de modelagem e simulação [5].

A HLA define um conjunto de regras que devem ser seguidas por todos os participantes da simulação integrada, chamados de federados. Os federados se comunicam pelo *Runtime Infrastructure* (RTI) e usam o *Federation Object Model* (FOM), que descreve os dados que serão trocados. O FOM pode ser interpretado como a "linguagem da federação".

É comum utilizar FOMs padronizados, chamados de *Reference FOMs*, e estender suas classes para atingir os objetivos de projeto particular [6]. *Reference FOMS* diminuem o tempo de desenvolvimento e de teste de um projeto, devido à reutilização de código.

O *Real-Time Platform Reference Federation Object Model* (RPR FOM) [7] é um FOM comumente usado em simulações de defesa, uma vez que apresenta a implementação de algumas classes importantes nesse cenário, como aeronave, radar e explosão. O RPR FOM 2.0 foi usado na solução proposta para interligação dos simuladores do COMAER via HLA.

III. Solução Desenvolvida

A solução desenvolvida buscou a interoperabilidade entre os simuladores da aeronave A-29 do COMAER. Esses simuladores apresentam diversos módulos, sendo dois deles primordiais para a elaboração da solução de interoperabilidade. O primeiro é o núcleo da simulação, que recebe as informações de todos os controles e as processa a fim de determinar a próxima posição da aeronave. O segundo é responsável por exibir a visualização do mundo ao piloto, chamado de *Out-The-Window* (OTW). O núcleo da simulação envia a posição atual da aeronave ao OTW, que calcula qual informação deve ser exibida, conforme arquitetura exibida na Fig. 1.

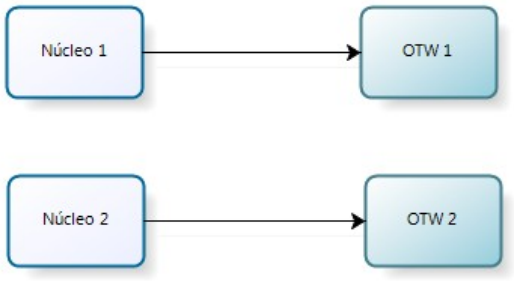

Fig. 1. Arquitetura atual utilizada nos simuladores da aeronave A-29.

Arquitetura da Integração via HLA

Para que o simulador existente fosse compatível com HLA, a fim de proporcionar, por exemplo, a interoperabilidade entre dois simuladores, desenvolveu-se o módulo "*bridge* HLA". Esse módulo exerce o papel de federado na simulação conjunta. Ele recebe os pacotes com informações do núcleo da simulação por meio de uma conexão *socket* e publica as informações de posição e orientação da aeronave no RTI.

A *bridge* assina para receber as informações dos demais federados. Ao receber as informações dos demais, ela envia pacotes com a informação de todas as aeronaves a OTW via *socket*. Os pacotes são enviados toda vez que uma nova informação do núcleo é recebida. Contudo, não é possível garantir que as posições das demais aeronaves também serão recebidas no mesmo momento. Dessa forma, essas informações devem ser interpoladas, baseadas na última atualização disponível. O processo de interpolação será detalhado no tópico "*Dead Reckoning*".

Além da interoperabilidade entre os simuladores, a arquitetura também permite a integração com outros federados compatíveis com HLA. Nesse sentido, utilizou-se o VR-Forces [8] para adicionar forças geradas por computador (*Computer Generated Forces* - CGF) à simulação integrada. As CGFs permitem incorporar na simulação entidades que tentam modelar o comportamento humano de modo que ações sejam tomadas automaticamente, sem a necessidade do homem na simulação [9]. Essas entidades podem representar aviões, radares, soldados, entre outros. Cada entidade pode ter um plano definido, como seguir outra entidade da simulação ou percorrer determinada rota.

A integração da solução proposta para interoperabilidade dos simuladores A-29 com o VR-Forces foi bem simples, pois este foi desenvolvido para suportar o RPR FOM 2.0. Dessa forma, as mensagens enviadas e recebidas pelo módulo CGF já estavam no padrão entendido pelo restante da simulação. A arquitetura completa é exibida na Fig. 2, contudo outros módulos compatíveis com HLA também poderiam ser integrados.

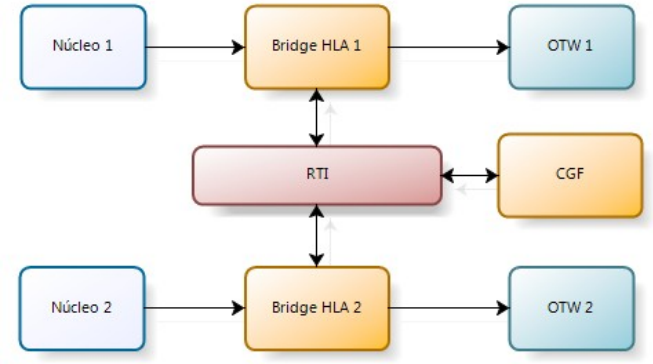

Fig. 2. Arquitetura proposta na interoperabilidade dos simuladores A-29.

Como todo o desenvolvimento da solução foi realizado no CCA-SJ, onde não há simulador do A-29, foi necessário encontrar uma alternativa para obter os dados do núcleo da simulação. No simulador real, a atuação do piloto sobre pedais, manche e manete fornecem informações ao núcleo. O núcleo processa essas informações com dados de aerodinâmica da aeronave, a fim de determinar a próxima posição. Para realizar a mesma tarefa sem a necessidade de utilizar o simulador real, usou-se o XPlane 10 [10] e um *joystick*. O XPlane recebia os comandos do *joystick* ou do teclado do computador e os aplicava na aerodinâmica de uma aeronave específica, devolvendo a próxima posição. Essa posição foi processada a fim de construir pacotes exatamente iguais aos que são enviados pelo núcleo do simulador do A-29, de forma a simular perfeitamente o módulo núcleo.

O módulo OTW não precisou ser simulado, visto que uma versão do seu software estava disponível no CCA-SJ.

É importante observar que essa arquitetura é escalável e, embora tenham sido integrados apenas dois simuladores e um CGF, seria possível colocar mais simuladores integrados.

Outros detalhes da implementação serão discutidos nos demais tópicos desta seção: "*Tecnologias Envolvidas*", "*Sincronização*", "*Dead Reckoning*" e "*Desafios*".

Tecnologias Envolvidas

As principais tecnologias utilizadas na solução de interoperabilidade foram:

- XPlane versão 10.1.0.14 [10]: utilizado para simular o núcleo da simulação;
- MÄK RTI 4.4.1 [11]: utilizado como RTI na integração via HLA;
- VR-Forces 4.3 [8]: utilizado como CGF;
- Eclipse Kepler [12]: utilizado para desenvolvimento do módulo "*bridge* HLA", em Java.

Sincronização

Um dos grandes desafios de integração de simuladores é a sincronização do tempo, uma vez que cada federado pode ter o seu tempo lógico local diferente. Esse tempo não tem unidade pré-determinada e um federado pode pedir para avançar o tempo em segundos ou minutos, por exemplo. Dessa forma, *a priori*, a federação não tem um tempo global sincronizado.

A RTI provê mecanismos para coordenar o tempo entre os federados. Aqueles que optem pela sincronização provida pela RTI podem ser do tipo *Time Regulating* e/ou *Time Constrained.* Federados que são *Time Regulating* podem enviar eventos com *timestamp*, enquanto os que são *Time Constrained* podem receber eventos com *timestamp*. Esses serviços podem ser habilitados para os federados através de chamadas *enableTimeConstrained* e *enableTimeRegulation* à RTI.

Embora exista toda essa infraestrutura de gerenciamento de tempo provida pela RTI, há um custo envolvido na sincronização, pois, sempre que um federado desejar avançar no tempo, deverá fazer uma requisição e então receber a permissão, quando todos os federados estiverem prontos. Isso cria uma sobrecarga de mensagens, uma vez que a publicação de informações poderia ser única. Dessa forma, uma alternativa é o uso de mensagens com *timeStamp* sem a utilização de federados que sejam *Time Regulating* ou *Time Constrained*. Embora isso dificulte o controle do tempo para pausar e simular em uma velocidade mais rápida, o intuito de diminuir a latência de comunicação entre os simuladores é atingido e essa foi a solução adotada. Vale lembrar que, caso interações de pausa ou simulação em escala sejam necessárias, isso pode ser implementado com mensagens customizadas, embora isso diminua a interoperabilidade dos simuladores.

Para usar mensagens com *timeStamp* a única alteração necessária é utilizar o método sobrecarregado da classe *RTIAmbassador,* o *updateAttributeValues* com a assinatura que tem como parâmetro o tempo lógico:

```
MessageRetractionReturn updateAttributeValues(
    ObjectInstanceHandle                theObject,
    AttributeHandleValueMap          theAttributes,
    byte[] userSuppliedTag,
    LogicalTime theTime)
```

Dessa forma, ao publicar alterações de um dado objeto, elas serão feitas com informação do tempo em que o evento ocorreu. Por outro lado, ao receber essa informação por meio do método *reflectAttributeValues*, é importante acessar a informação de *timeStamp* com o método:

```
public void reflectAttributeValues(
    ObjectInstanceHandle theObject,
    AttributeHandleValueMap theAttributes,
    byte[] userSuppliedTag,
    OrderType sentOrdering,
    TransportationTypeHandle theTransport,
    LogicalTime theTime,
    OrderType receivedOrdering,
FederateAmbassador.SupplementalReflectInfo
reflectInfo)
```

É importante observar que, como o tempo lógico não é padronizado na HLA, convencionou-se o uso do tempo atual em milissegundos, medido como a diferença entre o horário atual e a meia-noite de primeiro de janeiro de 1970 em UTC.

Uma outra dificuldade associada à sincronização dos computadores é que cada computador pode ter uma diferença de relógio, mesmo que de poucos milissegundos. Ainda que protocolos de sincronização de relógios, como o NTP [14], sejam utilizados, algumas dezenas ou centenas de milissegundos de diferença podem ser encontrados, de forma que não é recomendável confiar apenas no protocolo de sincronização de relógios. A solução encontrada foi, portanto, assumir que a diferença entre o tempo local e o tempo de um dado simulador será o tempo local subtraído pelo *timeStamp* da primeira mensagem recebida pelo simulador conforme evidenciado na Fig. 3.

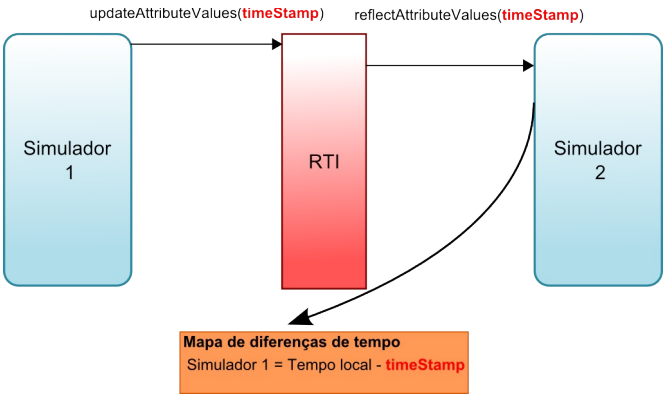

Fig. 3. Diagrama de fluxo para composição da diferença de relógios entre simuladores.

Percebe-se que, no primeiro envio de um dado evento que ocorreu no simulador 1, no tempo t_1, foi enviado o *timestamp* ao RTI por meio da chamada do método *updateAttributeValues*, o qual repassou o evento ao simulador 2 por meio do método *reflectAttributeValues*. Quando esse método é executado, cria-se uma entrada no mapa de diferenças de tempo, armazenando-se a diferença entre o tempo local e o tempo recebido no *timeStamp*. A partir desse momento, a diferença é o valor utilizado para as interpolações, conforme será descrito no tópico *Dead Reckoning*.

Dead Reckoning

Em geral, a taxa de atualização da renderização da OTW nos simuladores é de 60 Hz, no entanto, a taxa de comunicação das informações entre simuladores pode ser menor, implicando a necessidade de interpolar as posições e orientações de entidades remotas. O método padronizado de interpolação é conhecido como *dead reckoning* [13] e funciona de forma que cada simulador mantém um modelo interno das entidades remotas que ele representa. Além disso, a simulação mantém um modelo de *dead reckoning* dessa entidade. Pelo uso das interpolações, os simuladores não precisam reportar o status de suas entidades tão frequentemente.

O RPR FOM 2.0 [7] define quais algoritmos de *dead reckoning* são utilizados por meio do atributo *Spatial*, do tipo *SpatialVariantStruct*, conforme visto na Fig. 4, para todas as classes derivadas da *BaseEntity*, que é o caso das aeronaves.

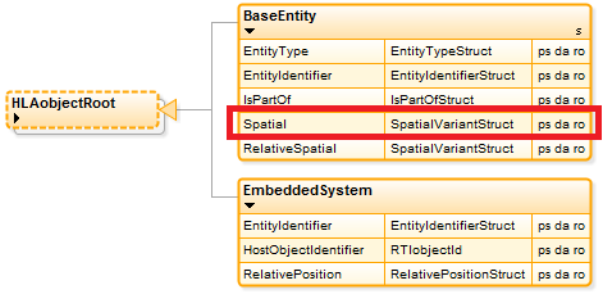

Fig. 4. Estrutura utilizada para definição do algoritmo de *dead reckoning*.

A estrutura *SpatialVariantStruct* é composta por um discriminante chamado *DeadReckoningAlgorithm*, o qual possui alternativas conforme descrito em [13].

Na solução adotada, utilizou-se a interpolação DRM FPW, ou seja, *Dead Reckoning Model (DRM)* com rotação fixa (F), variação de posição (P) e as coordenadas com referência no mundo, *world,* (W). Assim, a fórmula para a atualização da posição é definida na equação 3.

$$P = P_0 + V_0 \Delta t \quad (3)$$

em que P é a posição atual, em coordenadas geocêntricas, P_0 é a última posição recebida através do método *reflectAttributeValues*, V_0 é a última velocidade recebida também no mesmo método e Δt é o tempo atual, no simulador local, convertido para o tempo do simulador cuja entidade está sendo simulada - obtido pelo mapa de diferenças de tempos da Fig. 3 - subtraído do tempo de chegada do último *timeStamp*. Isso gera uma renderização mais fluida, porém, pode incorrer em pequenos erros uma vez que se supõe que a aceleração é nula na interpolação, o que não é verdade para muitas situações simuladas.

IV. Resultados

A solução implementada foi testada em dois computadores[1] em uma rede local. Um dos computadores executou os módulos relativos ao simulador 1 (núcleo 1, *bridge* HLA 1 e OTW 1), o módulo CGF e o RTI. O outro computador executou os módulos relativos ao simulador 2. Foram usados apenas dois computadores pois havia duas licenças *node-locked* disponíveis, contudo não haveria problema em usar mais computadores, por exemplo, separando o módulo CGF.

Foram realizados testes com diversas configurações, algumas delas serão detalhadas a seguir:

Núcleos de simulação com X-Plane

1 Máquina 1: CPU Intel(R) Xeon(R) E5606 2.13 GHz, RAM 12 GB, Placa de rede 1 Gigabit.
Máquina 2: CPU Intel(R) Core i7-4790 3.60 GHz, RAM 16 GB, Placa de rede 1 Gigabit.

Nessa configuração duas instâncias do software X-Plane [10] foram usadas para simular dois núcleos de simulação, conforme a arquitetura descrita na figura 2, com exceção do CGF. Para esse teste, pilotos da FAB avaliaram subjetivamente o tempo de resposta dos comandos de entrada e a qualidade visual percebida no OTW em situações de voo em ala. Em suas avaliações, perceberam que foi possível executar o voo, mas a imagem da outra aeronave estava descontínua quando muito próximas.

Como comentado, a interpolação das demais aeronaves da simulação foi feita com uso da equação 3, causando erro quando a velocidade não era constante. Considerando a taxa de envio de pacotes de 10 Hz e um voo conjunto sem grandes manobras, o erro médio observado entre a posição interpolada para uma aeronave e a posição correta recebida em um novo pacote é exibido na TABELA 1. Esse erro variou de acordo com a velocidade média da aeronave; para velocidades mais altas e considerando-se a mesma taxa de envio de pacotes, o erro de interpolação era maior.

TABELA 1: VARIAÇÃO DO ERRO MÉDIO DE INTERPOLAÇÃO COM A VELOCIDADE MÉDIA DA AERONAVE INTERPOLADA.

Velocidade (km/h)	Erro médio (m)
300	2,00
400	2,63
500	3,67
600	4,05

A TABELA 1 mostra que com a velocidade de 600 km/h (aproximadamente a velocidade máxima do A-29) o erro médio observado foi de 4,05 m. Esse erro gera um pequeno salto no posicionamento da aeronave, conforme Fig. 5, que chega a ser perceptível em voos em ala, tornando-se ruim para o treinamento. Quanto mais distante as aeronaves se posicionam, os saltos se tornam menores, até se tornarem imperceptíveis, quando o erro tem uma representação na projeção ortográfica menor que a largura de um pixel. Verificou-se experimentalmente que, em voos com as velocidades de 300 a 600 km/h, o erro torna-se imperceptível se a distância das aeronaves é maior que 180 m.

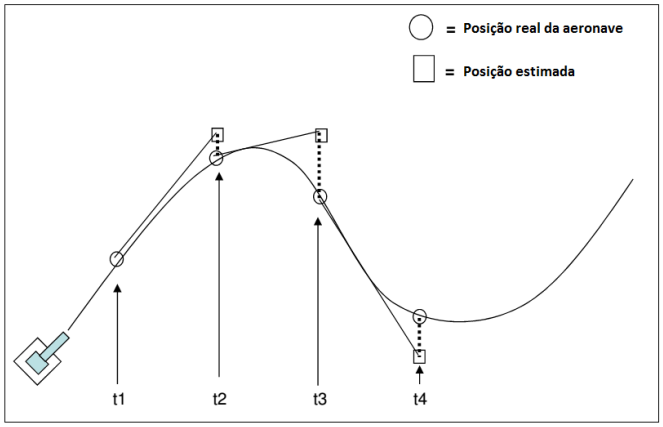

Fig. 5. Diferença entre as posições estimadas com o dead reckoning e as posições reais da aeronave (redesenhado de [15]).

Como solução para os saltos observados na imagem interpolada, pretende-se implementar uma solução de *smoothing*, conforme [16]. Outra opção é aumentar a taxa de envio de pacotes, que faz com que a atualização da posição correta da aeronave ocorra com mais frequência.

Pode-se ver um *screenshot* do voo em ala nessa configuração na Fig. 6.

Fig. 6. *Screenshot* de voo em ala, conforme visto pelo OTW do simulador do A-29.

Integração com CGF COTS

Uma das grandes vantagens da integração por HLA, se comparada com integrações *ad hoc*, é a possibilidade de integrar simuladores que não foram projetados para funcionarem em conjunto, desde que utilizem *Federation Object Model* comum. Como todo esse estudo foi baseado no RPR FOM 2.0 [7], conforme discutido na introdução, foi possível conectar os simuladores de voo com o simulador tático VR-Forces [8]. Algumas interações testadas foram a de colocar uma aeronave simulada como escolta, criando interações de tiro, de acordo com a detecção de uma aeronave inimiga, as quais ocorreram com sucesso. Uma outra facilidade testada foi a de fazer uma aeronave simulada seguir uma rota específica, simulando o comportamento de um líder, possibilitando o treinamento de voo em ala com uma entidade simulada. Essa configuração pode ser visualizada no *screenshot* do VR-Forces [8], conforme Fig. 7.

Os resultados apresentados mostram que a solução desenvolvida conseguiu criar a *bridge* para ligar os módulos do simulador do A-29 a outras entidades via HLA. Ainda não foi possível aplicar a solução implementada nos simuladores reais, contudo já foram realizados testes no laboratório existente nas dependências do CCA-SJ, que possui os mesmos softwares do simulador.

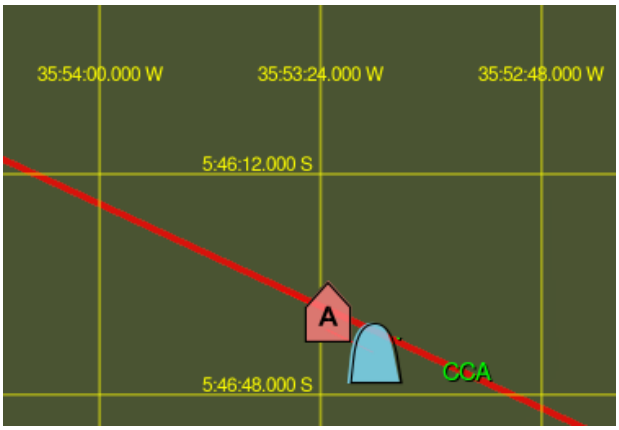

Fig. 7. Software de simulação tático VR-Forces integrado aos simuladores de voo. A entidade vermelha A segue uma rota preestabelecida enquanto a entidade azul é simulada em outra máquina da rede.

Como próximos passos, pretende-se aplicar a solução desenvolvida nos esquadrões de voo A-29, inicialmente no 2º/5º GAV, e verificar o impacto trazido pelo treinamento simulado de voos em pacote. Espera-se que a interoperabilidade reduza a demanda por horas de voo de treinamento na aeronave real e, consequentemente, aumente o *Cost Effectiveness Ratio* do simulador, resultando em grande economicidade para a nação.

V. CONCLUSÃO

Esse trabalho mostrou o desenvolvimento de uma potencial solução para uma arquitetura escalável de integração de simuladores não projetados para serem interligados por HLA, a qual foi testada e analisada em parâmetros objetivos e subjetivos, em um ambiente de laboratório.

Foram propostas soluções de sincronização de tempo e *dead reckoning* padronizadas por HLA e pelo RPR FOM 2.0. Percebeu-se a limitação de pequenos saltos nas correções de *dead reckoning* para distâncias menores que 180 m, a qual deverá ser corrigida por *smoothing* em trabalhos futuros.

Essa arquitetura poderia ser replicável em outros simuladores da FAB e teria as vantagens de integração com ferramentas de CGF COTS. Percebeu-se que apenas o uso de HLA não é suficiente para a integração, mas que há necessidade de se definir um *Federation Object Model* padronizado pela indústria, como é o caso do RPR FOM 2.0.

AGRADECIMENTOS

A equipe do CCA-SJ agradece à empresa MÄK por fornecer duas licenças *node-locked* de um mês para o MÄK *RTI* e para o *VR-Forces*. A realização desse trabalho não seria possível sem essa parceria.

Faz-se outro agradecimento à chefia do CCA-SJ e à chefia da Subdivisão de Simuladores, bem como a seus outros integrantes, que apoiaram o desenvolvimento desse trabalho.

REFERÊNCIAS

[1] J. M. Rolfe e K. J. Staples, “Flight Simulation”, Cambridge University Press, 1988.

[2] S. N. Roscoe, “Incremental Transfer Effectiveness”, Institute of Aviation, University of Illinois at Urbana-Champaign Savoy, 1971.

[3] S. R. Assis, “O Simulador de Voo nos Esquadrões da Aeronave A-29 Super Tucano da Força Aérea Brasileira”, Escola de Comando e Estado-Maior da Aeronáutica, 2013.

[4] IEEE: “IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)”, IEEE Std 1516-2010, IEEE Std 1516.1-2010, and IEEE Std 1516.2-2010, Disponível em: www.ieee.org, Acesso em: jun. 2015.

[5] D. Fullford, “Transitioning from DIS to the High Level Architecture (HLA)”, American Institute of Aeronautics and Astronautics, 1997.

[6] B. Möller, A. Dubois, P. Le Leydour e R. Verhage, “RPR FOM 2.0: A Federation Object Model for Defense Simulations”, Proceedings 2014 Fall Simulation Interoperability Workshop, 14F-SIW-039, Simulation Interoperability Standards Organization, 2014.

[7] SISO: “Real-time Platform Reference Federation Object Model (RPR FOM) Version 2.0”, Disponível em: http://www.sisostds.org/DigitalLibrary.aspx?EntryId=35932, Acesso em: jun. 2015.

[8] VR-Forces: Computer Generated Forces and Simulator Development, Disponível em: http://www.mak.com/products/simulate/vr-forces.html, Acesso em: jun. 2015.

[9] Modelling and Simulation (M&S) Master Plan, US Department of Defence, Out. 1995.

[10] XPlane versão 10, Disponível em: http://www.x-plane.com/desktop/home/, Acesso em: jun. 2015.

[11] MÄK RTI 4.4.1 Release Notes, Disponível em: http://www.mak.com/relnotes/RTI4.4.1ReleaseNotes.pdf, Acesso em: jun. 2015.

[12] Eclipse Kepler, Disponível em: http://www.eclipse.org/kepler/, Acesso em: jun.2015.

[13] DIS Steering Committee, “IEEE Standard for Distributed Interactive Simulation-Application Protocols.”, IEEE Standard 1278.1-1995(R2002), Disponível em: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6389876, Acesso em: jun. 2015.

[14] D. L. Mills, “Computer Network Time Synchronization-the Network Time Protocol”, CRC Press, 2006.

[15] P. A. Fishwick, “Simulation Model Design Execution”, 1995.

[16] K. Lin, M. Wang e J. Wangt, “Smoothing of Dead Reckoning Image in Distributed Interactive Simulation”, J. AIRCRAFT, VOL. 33, Nº 2, 1996.