basAR: Ferramenta de Autoria de Realidade Aumentada com Comportamento

Christopher Shneider Cerqueira
Departamento de Matemática e Computação - DMC
Universidade Federal de Itajubá - MG
Itajubá - BR
christophercerqueira@gmail.com

Claudio Kirner
Departamento de Matemática e Computação - DMC
Universidade Federal de Itajubá - MG
Itajubá - BR
ckirner@gmail.com

***Resumo*—As aplicações mais modernas de realidade aumentada estão tendendo a colocar objetos virtuais inteligentes no cenário de forma a reagirem aos usuários e as condições do ambiente, no entanto, faltam estratégias e ferramentas para facilitar a autoria desse tipo de aplicação. Este artigo trata da forma de organizar uma aplicação de realidade aumentada, separando em camadas de Infraestrutura, Atuação, Comportamento, Estrutura e Conteúdo. Além disso, propõe estruturas externas para configurar algumas destas camadas para que o desenvolvedor possa ajustar a aplicação quanto a sua necessidade.**

Realidade aumentada; ferramenta de autoria; organização da aplicação

I. INTRODUÇÃO

O desenvolvimento de realidade aumentada [1] ainda é complexo devido à dificuldade de produzir-se as partes necessárias para uma aplicação, incluindo os artefatos escolhidos para interação, o tipo de infraestrutura, os modelos 3D ou sons para serem utilizados.

No desenvolvimento da tecnologia até o usuário final, várias camadas de encapsulamento ocorrem, reduzindo a complexidade de desenvolvimento e separando as atividades de cada grupo de atuação.

No SVR2011, foi apresentada pelo prof. Dr. Claudio Kirner, uma abordagem de prototipagem rápida de aplicações, na qual, estas camadas foram separadas, diferenciando o tipo de característica do desenvolvedor e suas atividades.

A camada de Ferramenta de autoria pode ser subdivida para mostrar como uma ferramenta de autoria pode dar mais flexibilidade no desenvolvimento da estrutura e conteúdo.

Assim, neste artigo, são apresentadas as camadas de manuseio de dados, que podem compor um aplicativo de realidade aumentada, e como elas interagem entre si, iniciando por como podem ser feitas as abstrações dos dados de diversas fontes de dados, uma definição das camadas de infraestrutura do sistema, objetos atuantes, pontos de ação da estrutura e conteúdo como modelos e sons e uma camada de comportamento. Apresenta-se uma ferramenta de autoria chamada *basAR* (*Behavioral Authoring System for Augmented Reality*) que foi desenvolvida sobre estas definições de camadas, visando a construção de um ambiente de autoria de procedimentos de montagem/desmontagem, quebra-cabeças, sistemas adaptáveis de aprendizado, etc, que possam ser reconfigurados por arquivos externos.

II. FERRAMENTAS DE AUTORIA

As atuais ferramentas de autoria dispõem de pouca ou nenhuma atividade comportamental. Alguns dos principais ambientes vistos foram o *BuildAR*[3], SACRA[4] e o *Unifeye Design* [5].

O programa BuildAR, desenvolvido pelo *HitLabNZ*, não possui nenhuma capacidade de controle de comportamento, podendo apenas visualizar modelos.

O programa SACRA, desenvolvido por Rafael Santin e o prof. Dr. Claudio Kirner, possui alguns comportamentos, como a possibilidade de movimentar pontos nas bases de referências e a possibilidade de alterar o modelo exibido neste ponto.

O programa Unifeye Design, desenvolvido pela *metaIO*, possui um conjunto de comportamentos que podem ser visualmente arranjados para criar o comportamento da aplicação de acordo com a relação de marcadores e ou posicionamento/orientação [6].

Porém, dentre estas ferramentas de autoria, o único que é gratuito é o SACRA, enquanto que para utilizar o BuildAR e o Unifeye são necessárias aquisições de licenças. Na Figura 1, é visto um comparativo, sobre o tipo de comportamento de cada ferramenta de autoria citada e o basAR.

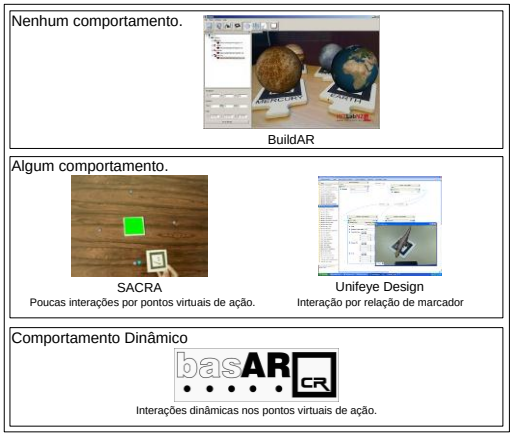

Figura 1 - Comparativo entre as ferramentas de autoria e o basAR, diferenciando os tipos possíveis de comportamento.

III. FONTES DE DADOS

Os dados das aplicações de realidade aumentada estão ligados ao acesso de hardware, sejam estas imagens de uma câmera, dados de sensores ou outros equipamentos. Contudo, para as interações, os dados que são importantes são referentes ao posicionamento, à orientação, à visibilidade e ao acionamento destes dados de hardware.

Assim, estes dados podem ser mapeados numa abstração que contém uma matriz de transformação que indica seu posicionamento e orientação de acordo com uma referência, uma matriz de referência para relacionar com a referência de base e um vetor de atributos de acordo com o mapeamento das ações da fonte de dados. A Figura 2 mostra uma representação desta abstração da fonte de dados.

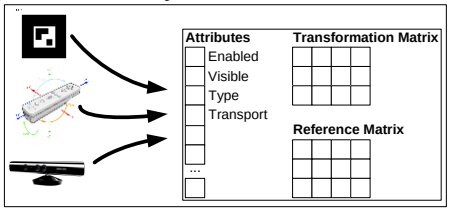

Figura 2 - Mapeamento do hardware em uma abstração para interação.

Atrás desta abstração podem ser colocadas diversas fontes de dados. Por exemplo, uma câmera controlada por um *framework*, do tipo do *ARToolKit* [7], retorna uma matriz de transformação da relação do marcador com a câmera e uma variável, indicando a visibilidade do marcador. Neste caso, pode-se considerar a matriz de referência zero, já que o valor retornado para a matriz de transformação é a relação entre o marcador e a câmera. Este passo de abstração é mostrado na representação da Figura 3.

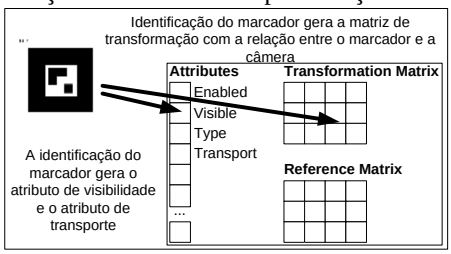

Figura 3 - Exemplo de mapeamento de um marcador do ARToolKit para uma fonte abstrata de dados.

Esta abstração diz respeito apenas entre as interações. Outras informações são importantes para cada tipo de dado, por exemplo: utilizando ARToolKit, é necessário informar o marcador a ser reconhecido; utilizando *Wiimote*[8], é necessário indicar o Endereço de Bluetooth, etc.

IV. APLICAÇÃO EM CAMADAS

Os dados que compõem o funcionamento de uma aplicação podem ser divididos de forma a facilitar a compreensão do desenvolvimento da aplicação, visualizado na Figura 4. Esta divisão pode ser em:

- Infraestrutura: responsável por especificar a área de trabalho (*workspace*) da aplicação, de onde a estrutura será montada.
- Estrutura: os posicionamentos e os tipos de modelos que estão sobre a infraestrutura.
- Contexto: modelos, sons e outros objetos definidos na estrutura.
- Atuação: o método como o usuário interage com a estrutura.
- Comportamento: definições das regras de interação entre os atuadores e a estrutura.

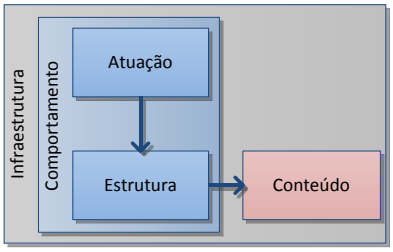

Figura 4 - Organização em camadas dos componentes da ferramenta de autoria BASAR

A. *Camada de Infraestrutura*

Infraestrutura é o conjunto das características do ambiente que são utilizadas para definir o posicionamento e a orientação dos componentes virtuais, criando uma área de trabalho, de forma a possibilitar a interação do usuário.

A infraestrutura lida com os dados, que caracterizam o ambiente e o relacionam com os atuadores, tratando os acontecimentos da aplicação, como, por exemplo, tratamento da execução do som, do modelo 3D, das interações, do mapeamento das fontes de dado, etc.

As configurações de infraestrutura dizem respeito aos itens que compõe o sistema como: componentes padrões do sistema, áudios de inicialização, erro padrão, o arquivo de relacionamento da estrutura com os atuadores e quantas infraestruturas e atuadores são utilizados. Estas configurações estão no trecho do arquivo de configuração da Figura 5.

```
Behavioral Authoring System for Augmented Reality
WINDOWED
VRML wrl/action/ballBlue.dat          # Holding
VRML Wrl/Action/ballGreen.dat         # Canwork
VRML Wrl/Action/ballRED.dat           # CannotWork
VRML Wrl/action/tampa.dat # Marker Cover
Audio/explosion.wav 0.5               # Error sound
Audio/backTrack.mp3 LOOP  0.3         # Backtrack
Audio/bell.wav      ONCE  0.5         # Opening Sound

Data/config_rules             # Rule Machine

# Bases
1                             # Number of bases
Data/config_base1             # Base configuration

# Actuators
1                           # Number of actuators
ARTKSM Data/transport   # Actuator ARToolKit Marker
```

Figura 5 - Arquivo config_basar, configuração de infraestrutura.

A área de trabalho e suas características são configuradas por outro arquivo (config_base, indicado na configuração de infraestrutura da Figura 6), como o trecho da Figura 6, que configura um uma referência do tipo marcador simples do ARToolKit, um som que pode ser tocado na visibilidade da área de trabalho e um som de erro.

```
# This file contains the setup for a workspace
BASE1                          # Workspace name

# Single ARToolKit Marker configuration
ARTKSM                         # Workspace source type
Data/Markers/base.patt         # Marker file
74.0                           # Marker width in millimeter
0.0 0.0                        # Marker center
USE_DEFAULT                    # Marker cover

# Base Sounds
Audio/bell.wav      ONCE  0.5 # Visible Sound
Audio/explosion.wav 0.5       # Error sound
```

Figura 6 - Trecho do arquivo config_base1, configurando a área de trabalho.

B. Camada de Estrutura

Estrutura é o conjunto de itens que estão sobre uma área de trabalho. Estes itens são pontos de ação (pontos de interesse ou pontos de interação). Este conceito foi introduzido pelo SACRA, onde estes pontos são regiões do espaço relativas à origem de uma área de trabalho que podem realizar alguma atividade. No caso do SACRA, estes podem ser movimentados e ou alternar modelos pré-programados.

No caso do aplicativo basAR, os pontos podem ter comportamento configurável na camada de comportamento, contudo seu posicionamento inicial, os objetos que são chamados e o raio de ação são configurados na camada de estrutura. Como é representado pelo trecho da Figura 7.

```
Pen                        # Point Name
DEFAULT_IPOINT             # ActionPoint Model File
Data/config_base1_pen      # OBJECT Model File
-200.0 -200.0 0.0          # ActionPoint translation
0.0 0.0 0.0                # ActionPoint rotation
1.0 1.0 1.0                # ActionPoint scale
900.0                      # ActionPoint action radius
```

Figura 7 - Trecho do arquivo config_base1, configurando pontos de ação da estrutura.

O desenvolvedor de estrutura configura o posicionamento, orientação e escala dos pontos de ação, quais e quantos objetos a aplicação vai suportar, definindo as características de cenário da aplicação.

C. Camada de Conteúdo

Conteúdo são os objetos utilizados pela aplicação. Nesta camada, ocorre o desenvolvimento temático da aplicação, na qual o conteúdo efetivo é adicionado, criando as representações lúdicas para o utilizador do aplicativo.

As definições da quantidade de objetos, tipo de objeto e o arquivo que descreve o objeto para ser carregado estão no trecho da Figura 8.

```
3                          # Number of objects

MODEL3D VRML Wrl/pen1.dat
MODEL3D VRML Wrl/pen2.dat
MODEL3D VRML Wrl/pen3.dat
```

Figura 8 - Arquivo config_base1_pen, configurando conteúdo de modelos 3D para um ponto de ação da estrutura.

Assim, sobre a área de trabalho definida na camada de infraestrutura, é montada uma camada de estrutura com os pontos de ação e cada ponto de ação pode ter um conjunto de modelos, como visto no esquema da Figura 9.

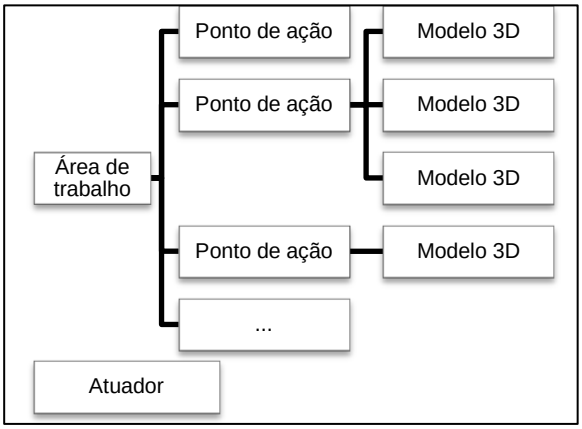

Figura 9 - Esquema da organização dos objetos das camadas de infraestrutura, estrutura e conteúdo.

D. Camada de Atuação

A camada de atuação inclui os artefatos utilizados pelo usuário para manipular o sistema, realizando interação com os pontos de ação da estrutura.

A camada de atuação, assim como a área de trabalho da infraestrutura, pode ser abstraída para a modelagem genérica de uma fonte de dados.

No caso da utilização de um marcador simples do ARToolKit, são necessárias as configurações do marcador, uma cobertura para o marcador, um modelo representativo para atuação (pá, mão, imã, etc.), e o posicionamento e o raio de alcance do ponto de ação. Na Figura 10, encontra-se um exemplo da descrição deste tipo de artefato de atuação.

```
# File describing the ARTKSM Actuator
ARTKSM1                        # Actuator name

# Single ARToolKit Marker configuration
Data/Markers/shovell.patt      # Marker
37.0                           # Width(mm)
0.0 0.0                        # Central
USE_DEFAULT                    # Marker cover
VRML wrl/Action/shovell.dat            # Symbolic model

# Point where the action occurs on the ARTSM Actuator
DEFAULT_IPOINT                 # Point model
20.0 0.0 0.0                   # Translation (x,y,z) (mm)
400.0                          # Action radius of the point
```

Figura 10 - Arquivo transport, definindo um atuador do tipo marcador do ARToolKit

E. *Camada de Comportamento*

Comportamento é a relação ou a interação entre os pontos de ação (atuador e pontos da estrutura), definido numa sequência de atividades pré-programadas.

Foi configurado um subgrupo de comportamentos para os pontos de ação:

- Estático: Ponto estático. (STAT, CHGST)
- Movimento: Possibilidade de arrastar um ponto. (DRGF, DRGRP)
- Atração: Atração controlada pelo sistema entre pontos. (ATTO, ATTRP, ATTA)
- Depósito: Atração ou repulsão controlada pelo usuário entre pontos. (DRPO, DRPA)
- Repulsão: Repulsão controlada pelo sistema entre pontos. (RPLO, RPLA)
- Configuração: Configuração de um ponto (TRA, ROT, SCL, CHGM, SETS, SETL, GETS, GETL)

Esquematizando as ações pela quantidade de pontos envolvidos é possível montar a separação da Figura 11, que divide as ações pela colisão do atuador livre com um ponto da estrutura, do atuador transportando um ponto da estrutura e colidindo com outro ponto e de ações de configuração.

Quando o atuador está livre, os pontos reativos são os que possuem ações de liberação de movimento e estático quando o atuador está transportando as possíveis ações são de atração, depósito e repulsão. As configurações não implicam numa ação do ponto, apenas em mudanças de atributos.

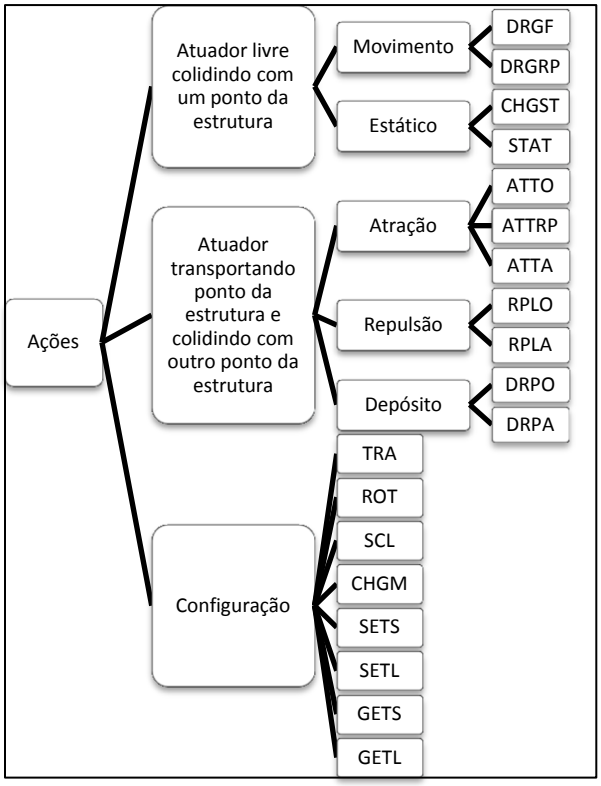

Figura 11 – Possíveis comandos no FARAC

As ações foram mapeadas em comandos, vistos na Tabela 1.

TABELA 1 – AÇÕES MAPEADAS EM COMANDOS

[BaseID] [PointID] [ViewMode] **STAT** <AUDIO> <OVER?>
[BaseID] [PointID] [ViewMode] **DRGF** <AUDIO> <OVER?>
[BaseID] [PointID] [ViewMode] **DRGRP** [NextState] <AUDIO> <OVER?>
[BaseID] [PointID] [ViewMode] **ATTO** [PointWaited] [NextState] <AUDIO> <OVER?>
[BaseID] [PointID] [ViewMode] **ATTRP** [PointWaited] [NextState] <AUDIO> <OVER?>
[BaseID] [PointID] [ViewMode] **ATTA** [NextState] <AUDIO> <OVER?>
[BaseID] [PointID] [ViewMode] **DRPO** [PointWaited] [NextState] <AUDIO> <OVER?>
[BaseID] [PointID] [ViewMode] **DRPA** [NextState] <AUDIO> <OVER?>
[BaseID] [PointID] [ViewMode] **RPLO** [PointWaited] [NextState] <AUDIO> <OVER?>
[BaseID] [PointID] [ViewMode] **RPLA** [NextState] <AUDIO> <OVER?>
[BaseID] [PointID] [ViewMode] **CHGST** [NextState] <AUDIO> <OVER?>
[BaseID] [PointID] **TRA** [X] [Y] [Z]
[BaseID] [PointID] **ROT** [X] [Y] [Z]
[BaseID] [PointID] **SCL** [X] [Y] [Z]
[BaseID] [PointID] **CHGM** [ModelToChange]
[BaseID] [PointID] **SETS**
[BaseID] [PointID] **SETL**
[BaseID] [PointID] **GETS**
[BaseID] [PointID] **GETL**

Uma maneira de organizar este comportamento é utilizando uma abstração de máquina de estado, onde em cada estado são definidas as características e comportamentos dos pontos de ação. A variação entre os estados, de acordo com os sucessos dos comportamentos dos pontos de ação, define a sequência das atividades. A Figura 12 contém um exemplo representando a montagem de duas peças (C e A) em duas posições (D e B), na qual a ordem de montagem não importa. Em cada estado, o sucesso de uma operação corresponde à mudança para outro estado.

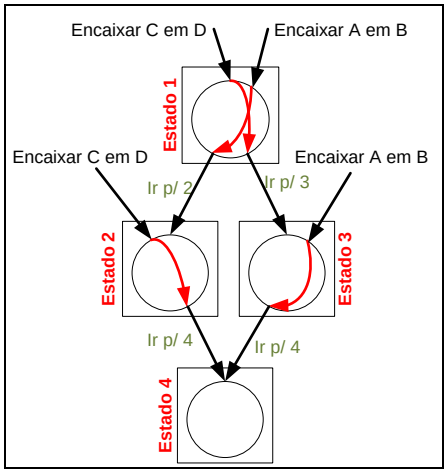

Figura 12 - Exemplo de uma máquina de estados de um encaixe de dois pontos (C e A) em duas posições (D e B)

A descrição do comportamento no formato de máquina de estado pode ser realizada como o trecho da Figura 13, que descreve o procedimento de montagem da Figura 12 utilizando comandos, que a cada estado serão aplicados aos pontos de ação. Na Figura 14 contém um comentário sobre os comandos utilizados para descrever os comportamentos.

```
BEGIN_STATE 1
        1 1 BOTH DRGF
        1 2 BOTH DRGF
        1 3 ONLY_BALL ATTO 1 3 Audio/bell.wav
        1 4 ONLY_BALL ATTO 2 2 Audio/bell2.wav
END_STATE

BEGIN_STATE 2
        1 1 BOTH DRGF
        1 2 ONLY_OBJECT STAT
        1 3 HIDE STAT
        1 4 ONLY_BALL ATTO 2 4 Audio/bell.wav
END_STATE

BEGIN_STATE 3
        1 1 ONLY_OBJECT STAT
        1 2 BOTH DRGF
        1 3 ONLY_BALL ATTO 1 4 Audio/bell.wav
        1 4 HIDE STAT
END_STATE

BEGIN_STATE 4
        1 1 ONLY_OBJECT STAT
        1 2 ONLY_OBJECT STAT
        1 3 HIDE STAT
        1 4 HIDE STAT
END_STATE
```

Figura 13 - Exemplo de como pode ser descrito o comportamento através de comandos.

1 BOTH DRGF - Os comandos deste formato:
[PointID] [ViewMode] DRGF indicam que o ponto (1), com o modo de exibição que mostra os pontos de ação e os modelos ativos (BOTH), pode ser movido livremente (DRGF – Drag Freely).

1 ONLY_BALL ATTO 2 3 Audio/bell.wav – Os comandos deste formato:
[PointID] [ViewMode] ATTO [PointWaited] <NextState> <SuccessAudio> indicam que o ponto (1), com o modo de exibição que só mostra o ponto de ação (ONLY_BALL), atrai o ponto específico (2) (ATTO – Attract Only) e na conclusão com sucesso da operação é direcionado um próximo estado (3) e executado um som de sucesso (Audio/bell.wav).

1 ONLY_OBJECT STAT– Os comandos deste formato:
[PointID] [ViewMode] STAT indicam que o ponto (1), com o modo de exibição que só mostra os modelos ativos (ONLY_OBJECT), está estático (STAT – Static).

1 HIDE STAT – Os comandos deste formato:
[PointID] [ViewMode] STAT indicam que o ponto (1), com o modo de exibição indicando que não está nada sendo exibido (HIDE), está estático (STAT – Static).

END_STATE – O comando indica o término do estado.

BEGIN_STATE – O comando indica o início do estado:
BEGIN_STATE [StateID]. Não pode haver dois índices de estado iguais na aplicação.

Figura 14 - Breve comentário sobre os comandos utilizados na Figura 12

V. ARQUIVOS DE CONFIGURAÇÃO.

Na seção 3, foram apresentados os arquivos de configuração para cada camada. Estruturando os arquivos temos um arquivo raiz (*config_basar*) que indica quais são os arquivos: de regras da máquina de estado (*config_rules*); quais os atuadores envolvidos e seu arquivo de configuração (no caso apenas *transport*); e quais as bases de referência da infraestrutura (no caso somente *config_base1*). O arquivo de base de referência da infraestrutura contém a configuração dos pontos de ação e seus modelos são listados num arquivo de descrição (*config_base1_n*), cada ponto pode conter vários modelos que são descritos em arquivos contendo seu formato (*model.dat*).

A hierarquia dos arquivos de configuração está representada na Figura 15, mostrando a maneira como a estrutura de dados é composta para formar a aplicação.

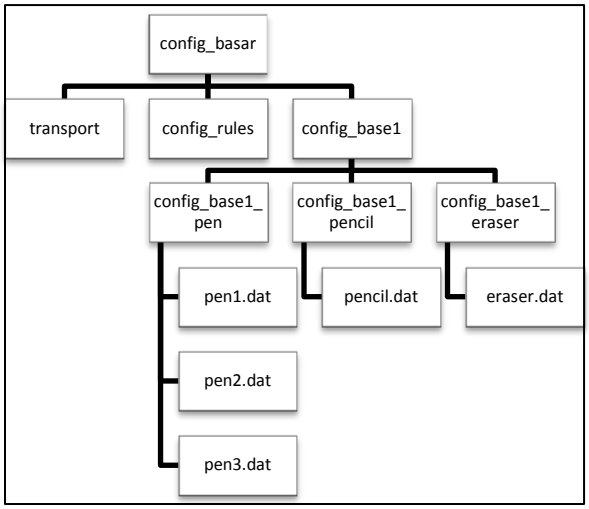

Figura 15 - Hierarquia dos arquivos de configuração.

VI. CONCLUSÃO

Este artigo demonstrou um software de autoria de realidade aumentada, em que são separadas as camadas de autoria em infraestrutura, estrutura, conteúdo, atuação e comportamento. Do modelo apresentado na Figura 1, são adicionadas as camadas de infraestrutura, comportamento e atuação na ferramenta de autoria, como visto na Figura 16.

A infraestrutura contém a organização do programa, indicando os arquivos de configuração de atuação, de base e das regras da máquina de estado e especificando a base de referência para exibição da estrutura. A estrutura é a organização dos pontos de ação que são as responsáveis por compor o "cenário" da aplicação. O conteúdo compõe-se dos objetos (modelos e áudio) que são inseridos na aplicação. A atuação é a especificação do artefato que o

usuário irá utilizar para interagir com os pontos de ação da estrutura. O comportamento é a forma com a qual as ações dos pontos são configuradas e o dinamismo de comportamento com a utilização de uma máquina de estado para "descrever" a característica e o comportamento do ponto num estado específico.

Também foi mostrado que os dados de infraestrutura e atuação podem ser abstraídos no software para uma estrutura padrão que será utilizada para testar a interação.

Os próximos passos para este trabalho são a expansão do universo de comportamentos possíveis, a utilização de outros modelos 3D, como *Collada*, e a utilização de outros artefatos de interação, como o Wiimote.

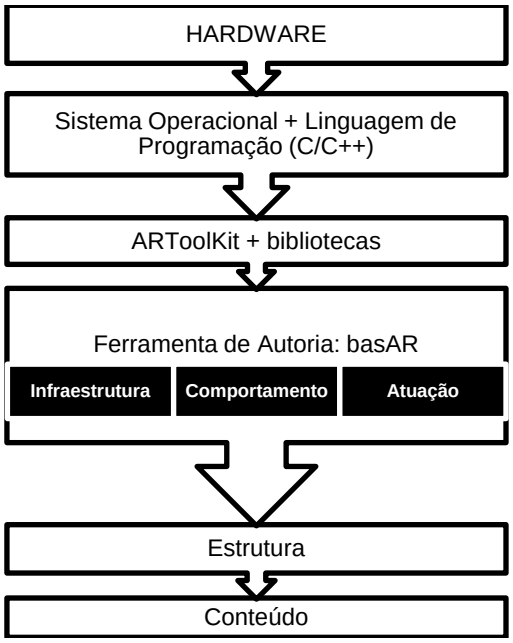

Figura 16 - Revisão do modelo de camadas, inserindo Infraestrutura, Comportamento e Atuação.

OBSERVAÇÕES

Este trabalho foi realizado no âmbito do Projeto "Ambiente Temático Interativo com Realidade Aumentada", financiado pelo CNPq (Proc. 58842/2009-7) e FAPEMIG (Proc. APQ-03643-10).

É possível realizar o download da ferramenta basAR nos sites dos autores:

https://sites.google.com/site/christophercerqueira/projetos/ear/basar

http://www.ckirner.com/basar

Nos sites também estão disponíveis exemplos e templates de comportamento prontos.

REFERÊNCIAS

[1] C. Kirner; R. Siscouto. "Fundamentos de Realidade Virtual e Aumentada". In: Kirner, C.; Siscouto, R.. (Org.). "Realidade Virtual e Aumentada: Conceitos, Projeto e Aplicações". 1 ed. Porto Alegre – RS: Sociedade Brasileira de Computação - SBC, 2007, v. 1, p. 2-21.

[2] C. Kirner. "Prototipagem Rápida de Aplicações Interativas de Realidade Aumentada. In: "Tendências e Técnias em Realidade Virtual e Aumentada". 1 ed. Porto Algre – RS. SBC, 2011, v.1 p. 29-54

[3] BUILDAR. BuildAR Pro 2.0. Disponível em: < http://www.buildar.co.nz/> Acesso em: 20 de julho de 2011.

[4] R. Santin, "Sistema de Autoria Colaborativa com Realidade Aumentada", Dissertação de Mestrado em Ciência da Computação, Universidade Metodista de Piracicaba, 2008.

[5] METAIO. metaio | Augmented Reality 3D. Disponível em: < http://www.metaio.com/> Acesso em: 20 de julho de 2011.

[6] E. R. Zorzal; A. Cardoso; C. Kirner; E. Lamonier Junior. "Técnicas de Interação para Ambientes de Realidade Aumentada". In: VI Workshop de Realidade Virtual e Aumentada - WRVA 2010, 2010, Santos - SP. Anais do VI Workshop de Realidade Virtual e Aumentada. Porto Alegre - RS : SBC, 2010.

[7] H. Kato; M. Billinghurst. "Marker Tracking and HMD Calibration for a Video-based Augmented Reality Conferencing System". In: IWAR'99, The 2nd Int. Workshop on Augmented Reality, San Francisco, USA, pp 85-94.

[8] J. C. Lee. "Wii Remote Projecs". Disponível em < http://johnnylee.net/projects/wii/>. Acesso em: 30 de agosto de 2011