Criação e Visualização de Interfaces do Usuário Cientes de Contexto para Realidade Aumentada com o framework VISAR

Allan Oliveira
Departamento de Computação
Universidade Federal de São Carlos
São Carlos, São Paulo, Brasil.

Regina Borges Araujo
Departamento de Computação
Universidade Federal de São Carlos
São Carlos, São Paulo, Brasil.

***Resumo* — A Realidade Aumentada (RA) é uma área de pesquisa que visa enriquecer a experiência de interação do usuário com o mundo que o cerca, entrelaçando elementos sintéticos, gerados pelo computador, com o mundo real, tornando cada vez menos perceptível a distinção entre o real e o virtual. Segundo pesquisadores da área de RA, o aspecto de interfaces de usuário na Realidade Aumentada é ainda uma área sub explorada, com vários desafios de projeto e implementação. Essas interfaces têm o potencial de usar elementos, por exemplo, 2d e 3d, adaptando-os a mudanças de contextos em tempo-real. Não há muitas ferramentas disponíveis para o desenvolvimento e visualização de interfaces de RA, muito menos que sejam cientes de contexto. Sendo assim, este artigo apresenta duas ferramentas que podem ser usadas de forma integrada para a criação e visualização de interfaces de RA cientes de contexto: um framework de visualização de interfaces de RA (VISAR) orientado a componentes de interface (padrões), que adapta, em tempo de execução, interfaces de RA em resposta a mudanças de contexto do ambiente; e um editor de interfaces de RA (VISAR-IE) que utiliza padrões para criar interfaces de RA cientes de contexto, permitindo o reúso e compartilhamento de componentes da interface. O uso de padrões acelera o tempo de projeto e implementação de interfaces de RA. Dois estudos de caso são também apresentados para validar o uso das ferramentas desenvolvidas: RA em mesa tangível multitoque, e um sistema de RA para bombeiros.**

Palavras Chave — Augmented Reality; Adaptive User Interface; Context Aware AR Interfaces;

I. INTRODUÇÃO

Realidade Aumentada (RA) é uma área de pesquisa em que objetos sintéticos gerados pelo computador suplementam e/ou compõem ambientes reais, fornecendo uma visualização de informações mais rica, presente no dia-a-dia e sem restrições de espaço, mesmo em situações de mobilidade do usuário.

De acordo com Azuma [1], três características principais definem a RA. São elas: combina objetos reais com virtuais em um ambiente real, oferece interatividade em tempo de execução e alinha (registra) elementos do mundo real com objetos virtuais.

Hoje em dia a RA está presente em uma grande variedade de áreas, como medicina, indústria, entretenimento e propaganda. O incremento no poder de processamento de dispositivos de visualização móveis, assim como, a diminuição destes, está aumentando ainda mais o alcance dessa tecnologia.

Os desafios para trazer a RA para o nosso dia-a-dia são tanto de hardware como de software, e incluem questões como o desenvolvimento de novos dispositivos de visualização, o manuseio da interação do usuário com objetos virtuais, a calibração em tempo real dos equipamentos, o rastreamento do usuário e objetos de interesse na cena, e o registro (alinhamento) de objetos virtuais no mundo real.

A complexidade dos desafios aumenta quando as interfaces de RA são cientes de contexto. Segundo Dey [2], contexto pode ser definido como qualquer informação que possa ser utilizada para caracterizar a situação de entidades (pessoa, lugar ou objeto) que sejam consideradas relevantes para interação entre um usuário e uma aplicação (incluindo o usuário e a aplicação). Para facilitar a captura e interpretação de contextos, podemos utilizar prédios inteligentes, que têm tecnologias que permitam sensoriar o ambiente. Além disso, tais prédios também podem ter instalado equipamentos de rastreamento dos usuários, tornando a RA uma interface ideal para ser utilizada em tais ambientes.

Para completar, sistemas de RA também possuem diferentes requisitos de Interface do Usuário (IU) em relação a sistemas tradicionais (como desktop, Web ou sistemas móveis). É uma interface do usuário diferente, já que sua maior parte está inserida no mundo real em 3d, embora a tela de visualização ainda possa ser usada para mostrar elementos 2d. Portanto, ferramentas e métodos para design existentes, devem ser complementados ou adaptados para servirem na criação e visualização de interfaces de RA cientes de contexto (ou novos criados), e pouco trabalho foi realizado nessa área específica [3].

Atualmente desenvolvedores de sistemas de RA tem uma preocupação maior com questões de hardware, como rastrear com precisão o usuário ou ter um dispositivo de visualização ótimo. Em consequência, as interfaces do usuário das aplicações de hoje, não demonstram o potencial completo da RA. Já no futuro próximo, quando tecnologias para implementar a RA estiverem mais consolidadas (como rastreamento), e sensores se tornarem comuns e abundantes nas casas, os desafios da RA serão mais voltados a interface do usuário. Nosso primeiro passo no suporte ao design de interfaces de RA são o framework VISAR e a ferramenta de autoria VISAR Interface Editor, descritos nesse artigo.

VISAR é um framework de desenvolvimento de RA que auxilia na criação de interfaces do usuário, através da utilização de padrões de interface (componentes gráficos implementados para solucionar problemas conhecidos) desenvolvidos para pré-

dios inteligentes (ambientes com o equipamento necessário para rastreamento do usuário e para sensoriamento do ambiente). Ele faz uma conexão entre IU e contextos, permitindo a adaptação da interface em tempo real de acordo com a situação do usuário.

Já o VISAR IE é um editor que permite aos desenvolvedores modelar interfaces de RA baseado nos padrões de interface e em regras de adaptação a contextos (ou cenários). Ele pode ser utilizado por um especialista na área da aplicação (comandante dos bombeiros, gestor de segurança da indústria, gerentes comerciais, etc.) que não tenha conhecimento em programação, mas exige um entendimento de ferramentas de modelagens (modelagem de fluxograma, modelagem de mapa mental) e interfaces de RA.

Em conjunto, o editor suporta o design de IU e seu comportamento enquanto o framework implementa e executa (em tempo real) a interface, sendo controlado pela aplicação. Para aqueles familiares com o padrão MVC (model-view-control), o framework seria o modelo e a visualização, e a aplicação seria o controle, que é responsável por atualizar os dados do modelo, que serão refletidos na visualização.

Alguns exemplos de aplicações de RA que o VISAR poderia ajudar na implementação seriam:

•Escritório: uma aplicação para auxiliar empregados em escritórios. Por exemplo, quando um trabalhador quer localizar seu chefe, a RA pode guiar o usuário através de um caminho virtual, ou quando o usuário manda imprimir um documento, ela pode iluminar a impressora utilizada.

•Casa: uma aplicação para auxiliar os moradores a fazer tarefas diárias. Por exemplo, quando o usuário quer cozinhar uma receita, a RA pode ajudar a achar os ingredientes mostrando rótulos virtuais nas portas de armários da cozinha onde cada ingrediente se encontra, ou ajudar a achar o controle remoto da televisão apontando uma seta virtual na direção dele.

•Emergência (bombeiros): uma aplicação para auxiliar bombeiros em ambientes de emergência. Por exemplo, a RA pode ajudar mostrando um modelo 3d de cada pessoa no ambiente, para que os bombeiros saibam onde cada um de seus colegas de time está, e onde estão as vítimas que já foram localizadas. Outro exemplo, seria um mapa 2d mostrando o que está acontecendo em cada quarto do local.

É possivel perceber que cada elemento da interface do usuário usado em uma dessas aplicações poderia ser utilizado nas outras também, sendo que estes elementos são os padrões de interface presentes no VISAR.

Continuando esta discussão, este artigo está organizado da seguinte forma: Seção 2 apresenta uma revisão bibliográfica da area de Realidade Aumentada. Seção 3 debate os trabalhos relacionados. O framework VISAR é introduzido na Seção 4. A ferramenta de autoria VISAR IE (Interface Editor) é descrito na Seção 5. Exemplos de aplicações são apresentados na Seção 6, seguido de conclusão (Seção 7) e referências bibliográficas.

II. REVISÃO BIBLIOGRÁFICA

A RA surgiu como um conceito que se completa à Realidade Virtual (RV), uma vez que considera o real como o ambiente que envolve o usuário, ao contrário da realidade virtual, em que o usuário é envolto por ambiente sintético e simulado. Como na RA não se tem o controle total do ambiente, ela é considerada mais complexa do que a RV, na qual o ambiente é mantido sob total controle [4]. Considera-se que essas duas áreas de estudo são correlatas, e que descobertas e pesquisas feitas em uma também afetam a outra.

Embora a RA tenha várias áreas de estudos repletos de desafios, como as áreas de displays, autoria, interação e usabilidade, o problema mais discutido e trabalhado pelos pesquisadores é o do registro (registration) [5]. Essa área envolve o rastreamento e calibração, sendo essencial para a existência da RA, por isso é a mais explorada.

O problema de registro está ligado ao alinhamento. Na RA o ambiente real se mistura com o ambiente virtual. Em outras palavras, o ambiente virtual deve encaixar-se no ambiente real perfeitamente, de forma que o usuário se sinta dentro de um ambiente misto. Para esse encaixe ocorrer de forma perfeita, é necessário que o alinhamento entre os objetos 3d e o mundo real seja exato.

Área / Tópicos	Artigos %	Citações %
Rastreamento / Localização	20.1	32.1
Interação	14.7	12.5
Calibração	14.1	12.5
Aplicações de RA	14.4	12.5
Display	11.8	5.4
Teste	5.8	1.8
RA móvel	6.1	7.1
Autoria	3.8	8.9
Visualização	4.8	5.4
Multimodal	2.6	0.0
Renderização	1.9	1.8

Tabela 1: Relação dos artigos e citações de acordo com a área em que se inserem, adaptado de [6].

Este trabalho está focado na área de interface do usuário da RA, no qual os desafios são, em sua base, os mesmos da IHC (interação humano-computador) tradicional. De acordo com Zhou et al. [6], este campo tem potencial para ser mais intensamente explorado (somente 8,6% dos artigos sobre RA são dedicados a essa área, somando autoria e visualização), mas atualmente o número de pesquisas tem aumentado.

Em relação à criação de interfaces do usuário e aplicações, existem duas abordagens principais para auxiliar desenvolvedores: frameworks, que trabalham em baixo nível, e ferramentas de autoria, que trabalham em alto nível.

Frameworks cobrem o processo completo de implementação de aplicações. Eles requerem que os desenvolvedores entendam e implementem em um baixo nível de programação, requerendo de aplicações. Eles requerem que os desenvolvedores entendam e implementem em um baixo nível de programação, requerendo mais tempo, mas permitindo que a IU seja projetada e controlada em detalhes. Eles normalmente oferecem funções para rastreamento e para gerar interfaces de RA básicas.

Os exemplos mais conhecidos são o ARToolkit [7] e o

Studierstube [8]. O primeiro oferece uma forma simples de desenvolver RA através de funções de rastreamento de marcadores quadrados. O segundo é uma solução mais completa, para desktop ou sistemas móveis, oferecendo componentes para renderização gráfica, vídeo, rastreamento, multimídia, armazenamento persistente e sincronização entre múltiplos usuários.

Também existem diversos frameworks comerciais, como por exemplo: Junaio, um browser de RA da Metaio; Unifeye, um SDK (Kit de Desenvolvimento de Software) de RA também da Metaio; Qualcomm AR SDK; Layar, um browser de RA com o foco na criação de diferentes camadas de visualização pelos próprios usuários. O intuito dessas ferramentas é oferecer diversas soluções de rastreamento (rastreamento de imagens, de objetos), especialmente para dispositivos móveis, com uma API (Interface de Programação de Aplicações) simples, mas potente, alvejando criação de conteúdo Web (especialmente de redes sociais).

Ferramentas de autoria permitem criar e editar conteúdo de RA através de programação visual. Elas têm um determinado número de padrões de interface implementados que o desenvolvedor pode trabalhar. Linguagens de descrição (ou linguagens de marcação) utilizam linguagens de alto nível de descrição para apresentar a IU em arquivos de texto.

Também existem diversos frameworks comerciais, como por exemplo: Junaio, um browser de RA da Metaio; Unifeye, um SDK (Kit de Desenvolvimento de Software).

Seichter et al. [9] introduz o composAR, uma ferramenta que foca na construção de aplicações de RA por leigos. Suas principais características são: permite a associação de objetos virtuais com reais (e definir interações para eles) e a possibilidade de usar scripts interpretativos.

Outra solução é o DART (Designer AR Toolkit), por MacIntyre et al. [10]. Construído no Macromedia Director, essa ferramenta permite criar aplicações de RA através de modelagem visual e scripting. Ela oferece vários comportamentos pré-definidos da interface, suporte de baixo nível a rastreadores, sensores e câmeras, e aceita a criação de aplicações de realidade virtual.

Uma limitação comum às ferramentas de autoria é que oferecem elementos de IU de RA muito básicos, faltando uma amplitude maior de componentes de interface. Além disso, são limitadas em controlar a aplicação em tempo real (normalmente, sendo necessário a construção de scripts para isso), e embora elas possam ser usadas para criar rapidamente aplicações, não ajudam na criação de aplicações complexas.

Nenhum dos trabalhos citados acima utiliza profundamente o conceito de padrões de interface, e quando são utilizados eles são poucos e simples, não incluem componentes 2d e 3d, e não permitem que desenvolvedores construam aplicações além de apresentações. Nosso trabalho tem como foco a interface do usuário, oferecendo uma solução completa para design e implementação da IU, enquanto ele permite que desenvolvedores mantenham controle total de suas aplicações.

III. UMA VISÃO GERAL DO FRAMEWORK VISAR

O framework VISAR e o editor VISAR IE fazem parte de um projeto maior em desenvolvimento no laboratório WINDIS (Wireless Networking and Distributed Interactive Systems), um dos grupos participantes do Instituto Nacional de Ciência e Tecnologia em Sistemas Embarcados Críticos (INCT-SEC).

Neste projeto, tecnologias de redes de sensores sem fio, robôs e veículos aéreos não tripulados são integrados para potencializar soluções atuais de segurança e proteção de infraestruturas críticas. VISAR e VISAR-IE fazem parte das soluções de interfaces avançadas, desenvolvidas como suporte à visualização em diferentes aplicações do projeto INCT-SEC.

A figura 1 mostra a visão geral da arquitetura do middleware de suporte às aplicações desenvolvido no projeto do INCT-SEC [11]. O círculo verde mostra a localização do framework VISAR, responsável pela visualização das interfaces de RA, cientes de contexto para as diferentes aplicações criadas. Uma rede de sensores, sem fio, coleta dados do ambiente. Estes dados são submetidos a um sistema de interpretação de contexto que entrega à aplicação (integrada ao framework VISAR) o estado de uma entidade. Esta informação é utilizada pelo VISAR para adaptar a interface de RA ao estado das entidades que interessam à aplicação.

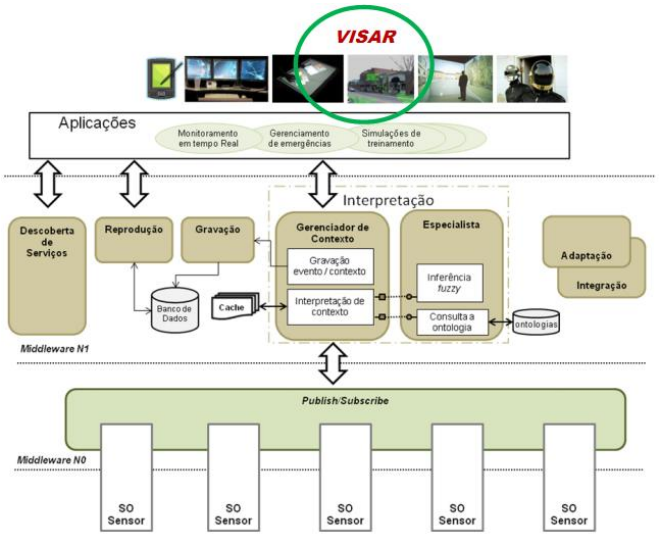

Fig. 1. Visão geral da arquitetura do middleware MIDSENSORNET (Projeto INCT-SEC) [11].

VISAR é um híbrido das abordagens de criação de IU, pois ele tem uma ferramenta de autoria (VISAR IE) para modelar a interface visualmente, esta ferramenta salva a interface em um arquivo com uma linguagem de marcação (XML), e este arquivo é lido pelo framework VISAR, que então carrega a estrutura da interface para que o programador possa alterá-la através de comandos durante a execução da aplicação.

Suas maiores contribuições são o modelo de padrões de interface e a junção das melhores vantagens de cada abordagem de desenvolvimento de IUs. O principal objetivo é facilitar o

desenvolvimento/implementação de interfaces do usuário de RA.

Em relação aos padrões de interface, eles são componentes de interface que representam soluções conhecidas e aprovadas para problemas comuns de design [12]. O exemplo mais comum de padrão de interface na RA é o marcador visual, que é a exibição de um objeto 3d em cima de um marcador, quando este marcador estiver visível para a câmera.

Aplicações que usam o VISAR, controlam os padrões em tempo real, preenchendo-os com a informação que se deseja mostrar ao usuário. Cada padrão tem sua API, que são funções usadas pela aplicação para gerenciar os padrões, seja alterando alguma informação, seja alterando a posição do padrão na cena.

IV. VISAR INTERFACE EDITOR

O editor VISAR-IE (Interface Editor) é uma ferramenta de autoria de RA, que funciona de forma integrada com o framework VISAR. Ele permite aos desenvolvedores de interfaces de RA focar na modelagem da interface e em suas regras. O editor é um software de autoria que suporta o projeto de interfaces de RA, por meio da escolha dos padrões que devem ser exibidos em cada cenário da aplicação, (isto é, situações em que o usuário ou outra entidade possa estar durante a execução da aplicação). Definição e exemplos de padrões são descritos na seção 5.1. O VISAR IE armazena o resultado do projeto da interface em um arquivo XML, que descreve a interface. Este arquivo é lido pelo framework VISAR (descrito na próxima seção), para carregar a estrutura da interface. O VISAR IE foi construído usando o plugin Graphical Editor Framework do Eclipse (Eclipse GEF).

O projeto da interface no VISAR IE pode ser feito em conjunto com um especialista na área da aplicação (ou por ele/ela próprio/a), pois a ferramenta não requer conhecimento em programação, mas exige um entendimento básico de ferramentas de modelagem, como fluxo de dados ou mapa mental, além de conhecimento sobre os padrões de interface de RA (não é necessário entender como eles funcionam, mas sim o que mostrarão para o usuário), e planejamento preciso da interface (uma boa análise de requisitos do sistema).

A figura 2 mostra a aparência do editor, e para explicá-la dividimos em quatro partes:

1) Paleta de padrões - onde ficam todos os padrões de interface disponíveis para serem usados;

2) Estrutura da interface em construção, com a exibição dos contextos e padrões que compõem a interface.

3) Tela de visualização do usuário. É possível ao projetista da interface fornecer o tamanho da tela (e resolução) nas propriedades. No caso da figura 2, a tela poderia ser de um celular, onde o desenvolvedor pode posicionar padrões 2d, indicando onde eles devem ser renderizados na tela, assim como seus respectivos tamanhos;

4)Propriedades do padrão selecionado. Cada padrão tem seu próprio conjunto de propriedades que pode ser alterado aqui. No caso da figura 2, as propriedades são referentes à tela de visualização do usuário.

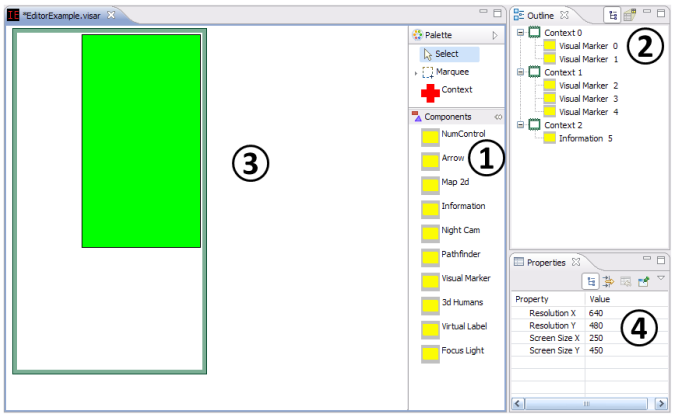

Fig. 2. VISAR Interface Editor.

V. O FRAMEWORK VISAR

A. *Padrões de Interface do VISAR*

Padrões de interface já são usados para projetar interfaces tradicionais, mas, até onde é de conhecimento do autor, ainda não houve tentativa de padronização de interfaces de RA de modo a facilitar a criação, o controle e o reúso de interfaces de RA.

De uma maneira geral, os padrões de RA podem ser classificados como estático ou dinâmico (referente a movimentação, como parado ou movimentando). Padrões estáticos são baseados em interfaces convencionais 2d, uma vez que, mesmo se tratando de interfaces de RA, ainda, é necessário exibir informações mais simples em 2d no visor do dispositivo do usuário. Já os padrões dinâmicos são formados por objetos projetados no ambiente, e são divididos pelo tipo de rastreamento necessário para funcionarem: um tipo para rastreamento genérico e o outro específico para visão computacional. Sendo assim, quatro tipos diferentes de padrões foram criados:

•Estático Normal: representa padrões que tem a posição fixa em relação ao usuário, normalmente 2d, e são fixados no display do usuário, por exemplo, no HUD (Head up Display). Suas propriedades são a posição e o tamanho na tela;

•Estático Rastreado: representa padrões com posição fixa em relação ao usuário, porém requerem informações de rastreamento do usuário e objetos de interesse na cena para funcionarem. Suas propriedades são as mesmas do tipo anterior;

•Dinâmico Rastreado: representam objetos virtuais que serão inseridos no ambiente real e, por isso, requerem um sistema de localização para serem registrados (alinhados no ambiente). Suas propriedades são: a posição (X,Y,Z) no ambiente, o seu tamanho e o ângulo de rotação;

•Dinâmico com Visão: representa padrões dinâmicos com rastreamento baseado em visão computacional (normalmente reconhecimento de fiduciais ou marcadores como o ARToolkit). Suas propriedades são as mesmas do tipo anterior mais a imagem do marcador.

Com essa classificação, é fácil estender a gama de padrões atuais criando novos. Para isso basta declarar o tipo de um novo padrão e parte/todas as funções básicas de funcionamento dele já estarão prontas.

Alguns exemplos de padrões podem ser vistos na figura 3. Estes são:

a) Rótulo Virtual: padrão dinâmico rastreado, que mostra um rótulo com informações sobre um determinado alvo (próximo ou no topo do alvo). Sua API tem as seguintes funções: estabelecer a posição (6DOF – 6 graus de liberdade, ou seja, tanto a posição quanto o ângulo de rotação) e inserir texto no rótulo;

b) Apontador: padrão estático rastreado, que mostra uma seta para guiar o usuário na direção do seu destino. Esta é uma seta 3d, portanto também pode apontar para cima e para baixo. O padrão deve saber a posição do usuário e o ponto de destino para, então, poder calcular a direção em que a seta deve apontar automaticamente e em tempo real. Sua API tem somente a função de estabelecer o ponto de destino;

c) Caminho Virtual: padrão dinâmico rastreado, que renderiza um caminho virtual que mostra uma rota para o usuário. Este caminho é construído através de um cálculo baseado no modelo 3d (CAD) do ambiente, e vai apenas da posição do usuário até a próxima esquina ou porta (para que assim ele não seja mostrado em locais fora da visão do usuário, sobrecarregando a tela com informações desnecessárias). Sua API tem somente a função de estabelecer o ponto de destino;

d) Marcador Visual: padrão dinâmico com visão (o mais comumente encontrado em interfaces de RA), que renderiza um objeto 3d no topo de um marcador. Sua API tem as seguintes funções: estabelecer a imagem do marcador e a posição (6DOF) do objeto 3d em relação ao marcador.

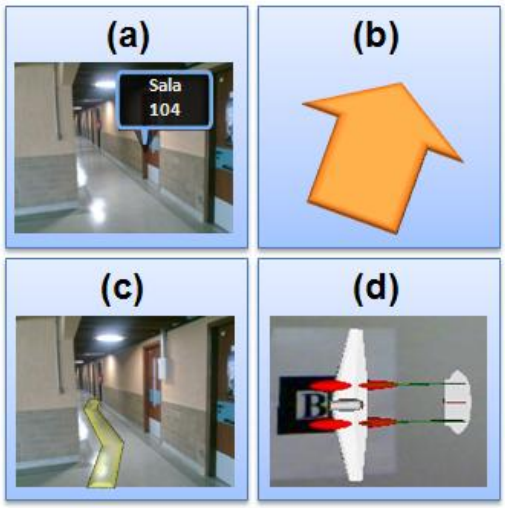

Fig. 3. Exemplos de padrões de interface.

Alguns outros exemplos de padrões de interface que poderiam ser criados seriam o humanóide 3d – uma representação 3d de um corpo humano inserido no ambiente para mostrar onde estão as pessoas próximas do usuário, a luz de foco – uma luz virtual para ser jogada em cima de um objeto (real) de interesse, o controle numérico – um componente para mostrar um valor mensurado do ambiente (visto na figura 5) e o mapa 2d – uma planta baixa do ambiente em 2d (visto na figura 5).

Todas as funções relacionadas a posição de padrões dinâmicos são normalmente conectadas diretamente a um gerenciador de rastreamento que possa alimentar os padrões com a localização (e portanto não são mencionadas na API).

Por outro lado, a posição de padrões estáticos é normalmente controlada diretamente pela aplicação. Isto ocorre por que a posição de padrões dinâmicos é um ponto no ambiente real que deve ser rastreado, enquanto a posição de padrões estáticos é somente um ponto na tela do dispositivo.

Cada padrão tem uma funcionalidade única para a aplicação. Alguns têm uso similar, mas com diferente aparência e tipo, como por exemplo, os padrões "Apontador" e "Caminho Virtual" (fig. 4 (b) e (c)) que servem de guias para o usuário, porém o "Caminho Virtual" necessita de um rastreamento muito preciso para funcionar. Outros padrões que tem aparência similar, mas uso diferente, como o "Mapa 2d" e o "GPS", onde o primeiro serve para ambientes internos e o último para externos.

Utilizando estes conceitos, qualquer componente gráfico novo que solucione um problema comum de interface pode se tornar um padrão de interface e, se desenvolvido em uma estrutura padronizada (como o modelo aqui proposto de padrões de interface), este componente pode ser reutilizado por outros, poupando tempo de desenvolvimento.

B. Funcionamento do Framework VISAR

O funcionamento do VISAR inicia com o carregamento, pela aplicação, do documento XML criado pelo VISAR IE, que contém a descrição da interface (os padrões e cenários que pertencem à interface). A aplicação envia para o VISAR o nome do dispositivo em que a interface deve ser mostrada e o VISAR renderiza os padrões do cenário inicial na tela.

O VISAR suporta a aplicação com funções para controlar a interface (APIs). O caminho dos dados, da aplicação para o VISAR, e depois para a renderização no display do usuário, pode ser visto na figura 4.

Após a renderização do primeiro cenário, o VISAR espera por um de dois tipos de comunicação por parte da aplicação: ativação ou desativação de um cenário ou alteração no conteúdo de um padrão (atualiza informações na interface).

Os cenários ativos têm todos os seus padrões renderizados, portanto, ativar um cenário faz o VISAR renderizar seus padrões, desativar faz o oposto. Assim, é possível mudar a interface em tempo de execução de forma fácil e rápida, desde que o projeto da interface tenha previsto o conteúdo a ser mostrado para o usuário em cada cenário.

Já a alteração no conteúdo de um padrão acontece quando a aplicação deseja atualizar o valor de alguma informação na IU. A figura 2 mostra um exemplo em que a aplicação detecta um aumento na temperatura de 60° para 85°. A aplicação, então, notifica o framework deste novo valor, através da API do pa-

drão (padrão do tipo Controle Numérico) atualizando a interface (tela da direita na figura 5).

Como a RA é baseada no rastreamento do usuário e na projeção de objetos virtuais no local correto, o VISAR deve estar conectado a um gerenciador de rastreamento, responsável por calcular a posição onde o padrão deve ser renderizado.

Cada padrão tem seu próprio requisito de rastreamento. O Gerenciador Interno do VISAR (Fig. 4) é responsável por registrar requisitos no gerenciador de rastreamento, para que o padrão seja alimentado com a localização atualizada, sobre a qual seus objetos virtuais devem ser renderizados.

Por exemplo, usando o ARToolkit como gerenciador de rastreamento, um padrão para renderizar um objeto 3d no topo de um marcador, deve receber a matriz de transformação necessária para isso, portanto, neste caso, (já que o ARToolkit não suporta os padrões se registrarem para receber informações de rastreamento) o Gerenciador Interno é responsável por obter esta matriz e passá-la para o padrão.

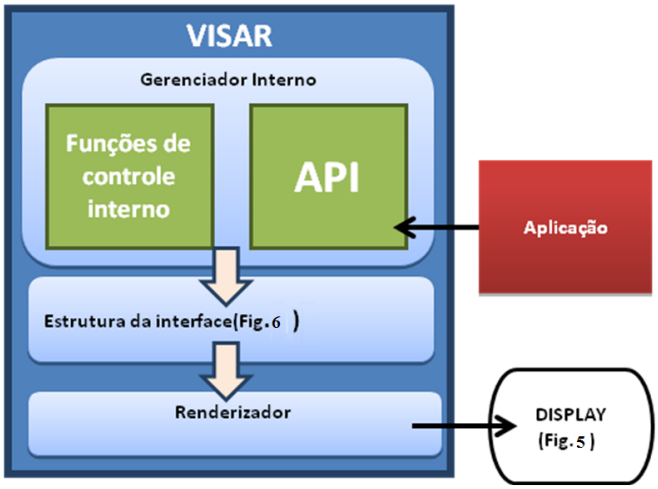

Fig. 4. Funcionamento e arquitetura interna do VISAR. Desde a aplicação mandando uma mensagem até o display onde a interface é renderizada.

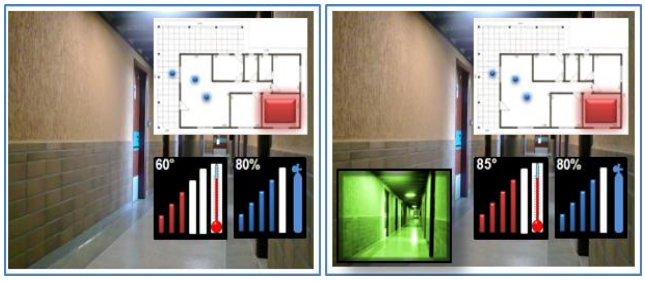

Fig. 5. Exemplo de interface criado pelo VISAR, em tempo real. Na tela da esquerda o cenário "NORMAL" está ativo; na tela da direita o cenário "ESCURO" está ativo.

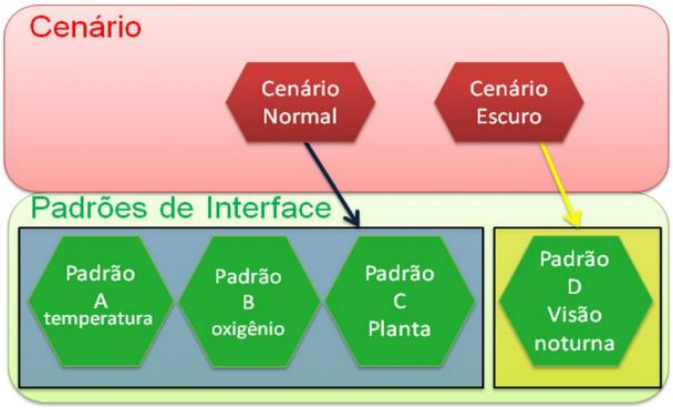

Fig. 6. Modelo da estrutura da interface contendo 2 cenários/contextos e 4 padrões.

Além do rastreamento, o Gerenciador Interno controla a sobreposição de padrões 2d. Se dois cenários estiverem ativos e ambos possuírem um padrão que fica na mesma posição na tela, o framework é programado para detectar esta situação e reposicionar o padrão com menor prioridade, por exemplo, utilizando o algoritmo de gerenciamento dinâmico de espaço de Bell e Feiner [13]. Essa prioridade é atribuída aos cenários no VISAR IE durante a fase de modelagem.

Um exemplo de aplicação simples, construída no VISAR, é um sistema para bombeiros em um ambiente de emergência. Esse sistema inicia no cenário "NORMAL", onde são exibidas para o usuário a temperatura da sala em que ele se encontra, a quantidade de oxigênio restante no galão e a planta baixa do local. Já, quando o usuário entra em uma sala totalmente escura, a aplicação, que é responsável por detectar a luminosidade, ativa o cenário "ESCURO" no VISAR, mostrando uma câmera de visão noturna. Além disso, quando a temperatura sobe de 60°C para 85°C, a aplicação notifica este novo valor ao VISAR, assim como também solicita que mais uma das barras do medidor seja pintada. Este exemplo, pode ser visto na figura 5, onde os padrões foram aumentados propositalmente para melhorar a visualização, e a estrutura da interface pode ser vista na figura 6.

O VISAR tem uma estrutura de dados que é uma lista de cenários, com cada um tendo sua própria lista de padrões, suportando, portanto, múltiplos cenários com múltiplos padrões. Os cenários seriam a modelagem de cada adaptação a diferentes contextos ou situações pelo sistema, cada um tendo sua "prioridade" e "estado". A prioridade pode ser usada para tomar uma decisão quando dois cenários ativos têm padrões se sobrepondo (2d) na tela. Já o estado serve para indicar se ele está ativo ou inativo.

Em testes feitos até então, só utilizamos câmeras conectadas

a um computador, mas um HMD (Head Mounted Display) está sendo providenciado. Um sistema de duas dimensões é usado para referenciar a posição na tela de objetos 2d, portanto, a troca de dispositivos de visualização é fácil de ser feita.

Esta implementação de padrões de interface é baseada no padrão MVC (Modelo-Visualização-Controle), em que a interface é dividida em três partes: o modelo (dados dentro de cada padrão); a visualização (modo que estes dados serão projetados para o usuário); e o controle (responsável por alterar o modelo, forçando assim uma atualização na visualização).

Internamente no VISAR temos o modelo e a visualização, assim a aplicação somente precisa fazer o papel de controle, acessando e alterando o modelo, o que faz o VISAR atualizar a visualização.

C. Arquitetura de Sistema do VISAR

A arquitetura do framework VISAR foi baseada no modelo de Reicher et al. [14] e pode ser vista na figura 7. Neste modelo, arquiteturas de RA são baseadas em seis subsistemas (criados a partir do padrão de arquitetura MVC). O trabalho apresentado se concentra principalmente no subsistema de apresentação, responsável pela disponibilização de informação para o usuário.

Os subsistemas de Rastreamento (Tracking), Modelo do Mundo (World Model) e Interação (Interaction) são representados na cor cinza, por não serem objetivos principais de desenvolvimento deste projeto. O subsistema de rastreamento recebe dados do Modelo do Mundo, correspondente à informação geométrica do ambiente real e virtual.

Após isso, diversas tecnologias podem ser aplicadas para rastrear a posição do usuário e objetos de interesse, e estas são fundidas para calcular a posição dos objetos virtuais. Essa informação é passada para o framework VISAR diretamente, como já foi dito anteriormente, para diminuir o atraso. O subsistema de Rastreamento também atualiza o modelo geométrico do mundo virtual, quando necessário. Existem várias implementações desses subsistemas, como o Ubitrack [15], para fundir dados de rastreamento dinamicamente.

O subsistema de Contexto (Context) tem a função de capturar e interpretar contextos do ambiente. O serviço de interpretação envia informação, já processada, para a aplicação, e utiliza informações do modelo real do mundo para informar corretamente onde cada evento está ocorrendo. Esse subsistema foi implementado no laboratório WINDIS [11].

O subsistema de Aplicação (Application) é onde está a maior parte da lógica dos desenvolvedores. Um aspecto importante a ser considerado na arquitetura apresentada, é que a aplicação recebe informações interpretadas e as utiliza através de duas funções do VISAR, que podem ativar ou desativar um cenário no VISAR para adaptar toda a interface ou atualizar as informações e conteúdos da interface.

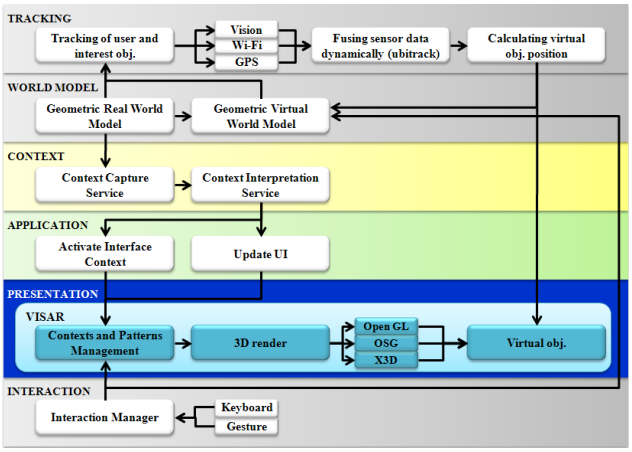

Fig. 7. Arquitetura de sistemas que utilizam o VISAR.

O subsistema de Interação é responsável por receber comandos de entrada do usuário, podendo ser transformados pelo seu gerenciador para funções da API do framework VISAR, atualizando a interface do mesmo modo da aplicação. Além disso, quando a entrada influencia a posição ou tamanho de um objeto virtual, ela também é repassada para o modelo de mundo virtual.

Por fim, o subsistema de Apresentação (Presentation) é onde o framework VISAR atua. As estratégias normais de apresentação foram substituídas por ele, que pode ser usado para criar, controlar e atualizar a interface do usuário.

D. Interação entre VISAR-IE e VISAR

A interação entre o VISAR-IE e o VISAR é feita através de documentos XML. Conforme já visto anteriormente, todos os padrões implementados no framework devem ter uma representação no editor, para que o projetista da interface possa utilizá-los.

Através do editor, a interface pode ser modelada, criando-se cenários para a aplicação e alocando os padrões a serem mostrados neles. Após projetar e configurar a interface no editor, esta é armazenada em um documento XML. O documento que contém a interface carrega todas as informações configuradas no editor, que são importantes para o VISAR, como por exemplo: uma lista de cenários com seus nomes, prioridades e padrões pertencentes a eles. Cada padrão apresenta um nome, tipo, descrição e propriedades como tamanho e posição na tela, dentre outras.

A figura 8 mostra um trecho de interface em formato de documento XML salvo no VISAR IE e que gerou a estrutura da figura 6. Este documento é carregado no VISAR, que armazena os dados em sua estrutura interna (fig. 6) e espera por chamadas de funções da aplicação com atualizações desses dados enquanto renderiza os padrões na tela.

```
<?xml version="1.0" encoding="UTF-8"?>
<context name="0">
   <priority>2</priority>
   <pattern name="Visual Marker 0">
      <properties description="airplane">
         <markername text="airplane.x3d"></markername>
         <size>10</size>
      </properties>
    </pattern>
   <pattern name="Visual Marker 1">
      <properties description="tank">
         <markername text="tank.x3d"></markername>
         <size>20</size>
      </properties>
    </pattern>
</context>
<context name="1">
   <priority>1</priority>
   <pattern name="Visual Marker 2">
      <properties description="flag1">
         <markername text="flag.x3d"></markername>
         <size>5</size>
      </properties>
    </pattern>
```

Figura 8: Arquivo XML que descreve a interface com cenários e padrões.

VI. ESTUDOS DE CASO

Duas aplicações serão dadas como exemplo, sendo a primeira mais simples e já implementada e a segunda mais complexa.

A. RA tangível em mesa multitoque

Esta aplicação foi implementada em uma mesa multitoque tangível, construída no laboratório WINDIS, como parte do projeto INCT-SEC. Nesta aplicação, o usuário deve controlar um avião para executar uma ação sobre um alvo. Na mesa, um mapa é colocado como tela de fundo, e mostra a posição inicial do avião. O usuário pode, então, controlar o caminho a ser sobrevoado pelo avião e que é identificado através de pontos chave (waypoints), sendo o último waypoint considerado o alvo nesta aplicação.

Os comandos na mesa (inserção de waypoints, ações e o alvo) são realizados através de objetos tangíveis (posicionam-se os objetos sobre o mapa na posição desejada). Como é possível redimensionar e mover o mapa, isto faz com que os objetos tangíveis saiam de suas posições originais. Para solucionar este problema, a RA foi utilizada para representar tais objetos, de forma que eles sejam usados somente como um "carimbo" (colocados e removidos) para determinar suas posições no mapa. Depois dos objetos serem retirados, eles são representados virtualmente, em 3D, através de RA (sendo necessário um HMD ou celular para visualização do virtual).

Fig. 9. Mesa tangível multitoque com RA.

Para rastrear a posição da mesa, quatro marcadores foram utilizados (um em cada canto), para a RA poder ser registrada corretamente sobre ela e gerar uma visualização 3D dos objetos de interesse. A figura 9 mostra esta aplicação rodando, onde pode ser vista a mesa com seus quatro marcadores, a interface da mesa (um mapa do ambiente), e um avião reproduzido através da RA pelo framework VISAR.

Nesta aplicação, assim que o usuário posiciona o objeto tangível, a mesa multitoque informa ao VISAR o ID do objeto tangível utilizado e a posição onde este objeto está na mesa. A cada atualização no mapa, caso ele seja movido ou redimensionado, uma nova posição do objeto é passada para o framework atualizá-lo. Caso um objeto seja retirado da mesa pelo usuário, por exemplo, um waypoint já ativado, a mesa informa ao VISAR qual objeto foi retirado. Cada objeto foi modelado como pertencendo a um cenário. Quando um objeto é utilizado ou desativado, o VISAR ativa ou desativa o cenário correspondente a ele.

B. Sistema de RA para bombeiros

Esta é uma aplicação de alto grau de complexidade (devido principalmente ao rastreamento, que não é foco deste trabalho) intencionada a auxiliar bombeiros em ambientes de emergência, como incêndios, resgates, mergulhos, salvamentos, etc. Vários cenários podem ser encontrados em cada uma das situações acima, portanto, a interface deve ser modelada para se ajustar bem em cada um.

Por exemplo, em um caso de incêndio, se ele está ocorrendo em uma sala fechada a um bom tempo, a porta do local não pode ser aberta, pois isso faria oxigênio entrar rapidamente, o que causaria uma explosão (conhecida como backdraft). Neste caso, a RA pode auxiliar mostrando que a porta não pode ser

aberta, ou quais portas não podem ser abertas, no caso de ser mais de uma.

Uma parte da interface desta aplicação pode ser vista na figura 10. Existem vários padrões de interface na figura para mostrar a utilidade da RA, mas, em uma situação real, eles são selecionados por cenários e poucos devem aparecer simultaneamente para não obstruir a vista do usuário ou sobrecarregá-lo com informações.

Os padrões de interface presentes na figura são listados abaixo:

•Uma lista de tarefas completadas e a serem feitas;

•Uma câmera de visão noturna (ligada somente no escuro);

•Um mapa 2d do ambiente, mostrando o local de ocorrência do fogo e também a localização de outros bombeiros;

•O valor da temperatura no local, assim como, a quantidade de oxigênio restante no galão do bombeiro;

•As horas ou um cronômetro com o tempo decorrido desde o começo da ação;

•A conectividade do equipamento com a rede sem fio que está alimentando a aplicação com os dados do ambiente;

•Um caminho virtual mostrando a rota até o destino;

•Um "X" na porta que não deve ser aberta;

•Uma representação 3d de pessoas (no caso vítimas) no ambiente e suas localizações.

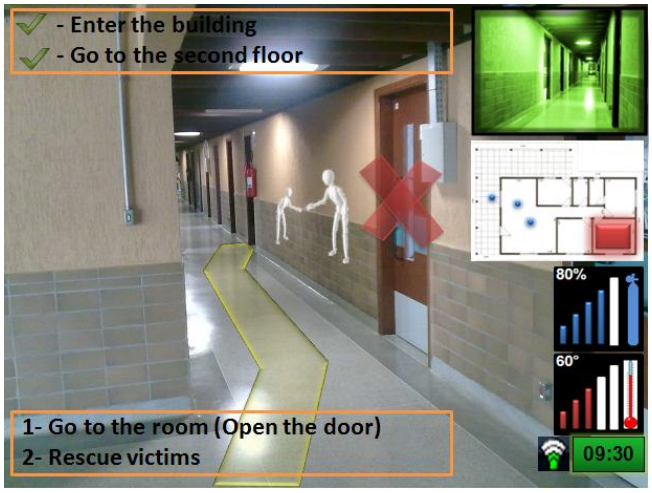

Fig. 10. Interface do Sistema de RA para bombeiros em ambientes de emergência.

VII. CONCLUSÕES E TRABALHOS FUTUROS

Um aumento no desenvolvimento de dispositivos e aplicações de RA está ocorrendo. Desafios de software e hardware estão sendo superados com dispositivos de visualizações menores, com maior processamento e mobilidade.

Um aspecto importante da RA é a dificuldade de construção de interfaces do usuário, uma tarefa que pode consumir uma grande quantia do tempo de desenvolvimento. As interfaces do usuário da RA devem ser capazes de se adaptar em resposta a diferentes contextos ou cenários, em tempo real, enquanto utilizam uma mistura de 3d com 2d para liberar todo seu potencial.

As abordagens a esta tarefa são basicamente: frameworks, que consomem grande quantidade de tempo, mas podem construir IUs bem detalhadas e flexíveis. Ferramentas de autoria, com alguns padrões de interface (componentes pré-implementados) prontos, mas são difíceis de serem estendidas. Linguagens de descrição, que descrevem a IU em uma linguagem de marcação, mas limitam o controle do desenvolvedor tem sobre a interface em execução.

Além disso, apesar da existência de um pequeno número de sistemas e soluções para RA ciente de contexto, estas soluções são limitadas a contextos tradicionais como localização ou iluminação.

Este artigo apresenta um framework de interfaces de usuário de realidade aumentada, com um editor para desenvolvimento ágil de interfaces adaptativas. Essas ferramentas foram criadas para diminuir a complexidade do desenvolvimento de interfaces de RA. Suas características principais são:

•Permitem o design de interfaces 3d projetadas no ambiente real e de interfaces 2d projetados na tela do dispositivo de visualização (que pode ser a viseira de um head mounted display);

•Possibilitam a modelagem de interfaces, direcionadas a cada cenário encontrado pelo usuário (contextos);

•Asseguram a rápida implementação e design de interfaces, ao construí-las baseadas em componentes implementados (padrões);

•Facilitam o reúso e compartilhamento de padrões entre desenvolvedores.

Elas são basicamente uma camada de apresentação/visualização em aplicações de RA, tentando lidar com as necessidades dos desenvolvedores em respeito a IUs de RA (desde o design até a implementação).

Como trabalhos futuros, planejamos expandir o editor VISAR IE, tornando possível posicionar, não somente padrões 2d na tela, mas, também padrões 3d na cena (no ambiente), adicionando o modelo 3d do ambiente no editor. Desse modo, o desenvolvedor vai poder ajustar a posição de padrões 3d, já na fase de modelagem (atualmente essa tarefa é realizada durante a fase de implementação).

Pretendemos estender o VISAR e aprimorá-lo, melhorando a conexão entre o gerenciador de rastreamento (talvez adotando um como fixo). Novos padrões também podem ser desenvolvidos, fazendo o VISAR atender a um maior número de aplicações.

REFERENCES

[1] R. Azuma, "A Survey of Augmented Reality", Presence: Teleoperators and Virtual Environment, vol. 6, no. 4, 1997, pp. 355-385.

[2] Dey, A. Abowd, G. "Towards a Better Understanding of Context and Context-Awareness", Em Workshop on The What, Who, Where, When, and How of Context-Awareness, as part of the 2000 Conference on Human Factors in Computing Systems (CHI 2000), 2000.

[3] J.E. Swan II, J.L. Gabbard, "Survey of User-Based Experimentation in Augmented Reality", in Proceedings of 1st International Conference on Virtual Reality, USA, 2005, pp. 22-27.

[4] Bimber, O. Raskar, R. "Spatial Augmented Reality Merging Real and Vir-tualWorlds", A K Peters LTD, 2005

[5] Ahad, R. Hossain, S. "Augmented reality and its challenges", SICE 2004 Annual Conference. vol. 2, Agosto 2004, pp. 1041 - 1043.

[6] F. Zhou, B. Duh, M. Billinghurst, "Trends in Augmented Reality Tracking, Interaction and Display: A Review of Ten Years of ISMAR", In Proceedings of ISMAR, 2008, pp. 193-202.

[7] H. Kato, M. Billinghurst, B. Blanding, R. May, "ARToolKit", Technical report, Hiroshima City University, 1999.

[8] D. Schmalstieg, A. Fuhrmann, G. Hesina, Z. Szalavári, L. M. Encarnação, M. Gervautz. "The studierstube augmented reality project", Presence: Teleoperators and Virtual Environments, vol. 11, 2002, pp. 33–54.

[9] H. Seichter, J. Looser, M. Billinghurst, "ComposAR: An intuitive tool for authoring AR applications", In 7th IEEE/ACM International Symposium on Mixed and Augmented Reality, 2008, pp.177-178.

[10] B. MacIntyre, M. Gandy, S. Dow, J. Bolter, "DART: A Toolkit for Rapid *Design* Exploration of Augmented Reality Experiences", In Proceedings of the 17th annual ACM symposium on User Interface Software and Technology, 2004, pp. 197–206.

[11] Beder, D. M., Araujo, R. B. "Towards the Definition of a Context-Aware Exception Handling Mechanism.", Workshop on Exception Handling in Contemporary Software Systems (EHCoS), SP, 2011.

[12] A. Seffah, "The evolution of *design* patterns in HCI: from pattern languages to pattern-oriented *design*", in Proceedings of the 1st International Workshop on Pattern-Driven Engineering of Interactive Computing Systems, 2010, pp. 4-9.

[13] Bell, B. Feiner, S. "Dynamic Space Management for User Interfaces", ACM Symp. on User Interface Software and Technology, 2000, pp. 238-248.

[14] Reicher ,T. MacWilliams , A. Brügge , B. Klinker, G. "Results of a Study on Software Architectures for Augmented Reality Systems", Proceedings of the The 2nd IEEE and ACM International Symposium on Mixed and Augmented Reality, 2003, pp. 274.

[15] P. Keitler, D. Pustka, M. Huber, F. Echtler, G. Klinker, "Management of Tracking for Mixed and Augmented Reality Systems", The Engineering of Mixed Reality Systems, Human-Computer Interaction Series, London, 2010, pp. 251-273.