

Computação Onipresente

**(UBICOMP - Ubiquitous
Computing)**

Manoel C. M. Neto
GSORT- IFBA

Criação de Sistemas Ubíquos

- Construir, Implantar e Manter Ubicomp exige MUITO esforço = CUSTO (em todos os sentidos)
- Por isso é importante PENSAR muito antes de meter a mão na massa...antes mesmo de projetar:
 - O que e porque eu quero construir?
 - O que eu quero aprender? O que vai acontecer depois que eu construir (suposição)?
- Pensar antes evita esforço desnecessário.
- Entender o que você está construindo ajuda a imaginar como alcançar os objetivos (comerciais ou científicos)

Motivações para Construir Ubicomp

- O que você deseja fazer com o sistema?
- Quem são os usuários (Desenvolvedores e Finais)?
- Qual a longevidade do Sistema?
- Tudo isso deve estar de acordo com as decisões de design e implementação.
- Algumas motivações incluem:
 - Projetar sistemas futurísticos para explorar (entender) ubiquidade na prática

Motivações para Construir Ubicomp

- Algumas motivações incluem:
 - Explorar empiricamente reações de usuários à Ubicomp
 - Obter dados reais para (talvez) resolver problemas computacionais.
 - Criar experiência que aumente o interesse em Ubicomp
 - Criar ambientes de testes em laboratório para estimular pesquisas futuras.
 - Explorar hipóteses que estejam relacionadas a tornar Ubicomp mais natural.
 - Testar os limites de tecnologias computacionais em ambientes Ubicomp.
 - Atender as necessidades percebidas em um domínio de problema ou questão social.

Motivações para Construir Ubicomp

- Algumas motivações incluem:
 - E você? O que lhe motiva a construir um sistema ubíquo?
- O primeiro passo é definir os seus objetivos.

Definir seus Objetivos

- É importante considerar onde será (deverá) alocado cada esforço.
- É importante definir quais partes do problema precisam ser totalmente implementadas
- Desses partes quais são "computáveis" para alcançar o resultado esperado.
- Ex.: Realizar estudos de caso em menor escala para avaliar (APENAS) a reação de usuários a sensibilidade de contexto demandaria grande esforço (para conseguir definir automaticamente um contexto confiável para o teste).
- Como resolver sem "gastar muito" ?

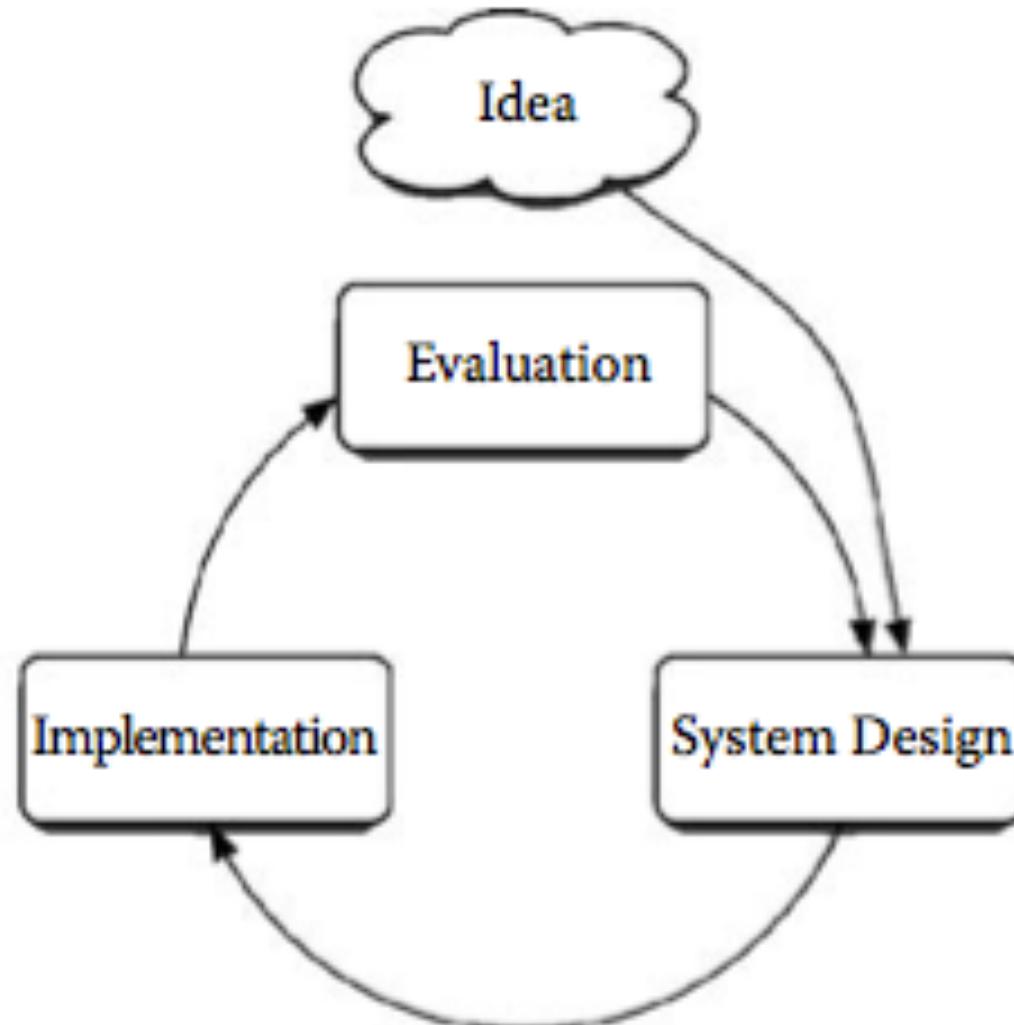
Definir seus Objetivos

- Vista sua roupa de sapo e dê seus pulos como se diz na Bahia!
- Alguém já ouviu falar na técnica do "Mágico de Oz" ?
 - Usa humanos para avaliar resposta de sistemas não implementados
 - O sistema tem que ser suficientemente simples para que o "Mágico" possa iludir os usuários (que não podem saber que o sistema não é real).
 - O "Mágico" estimula os usuários a usarem (inputs) o sistema observa as reações (outputs).

Definir seus Objetivos

- Técnica do "Mágico de Oz" por Timothy Griffin.
 - gather information about the nature of the interaction
 - test which input techniques and sensing mechanisms best represent the interaction (so that subsequent effort developing or adapting sensing technologies is appropriately directed)
 - test the interaction of a device before building a functional model
 - find out the kinds of problems people will have with the devices and techniques
 - investigate aspects of the products form such as visual affordance (whether the product shows how it can be used)

Definir seus Objetivos



Definir seus Objetivos

- Ambientes mais robustos como salas inteligentes têm custo maior (para construir) e dificilmente podem ser completamente simuladas com pouco esforço.
- Mantenha os olhos nos seus objetivos para garantir que o esforço seja recompensado.
- Mas seja flexível pois os seus objetivos podem mudar ao longo do tempo.
- Explorar protótipos pode permitir descobrir problemas para resolver e Hipóteses para testar Ajuda a definir seus objetivos
- Um protótipo que apenas aumente o interesse na área pode abrir os caminhos para a construção de um sistema muito maior e complexo.

Definir seus Objetivos

- Isso é uma consequência natural da exploração científica de um problema ubíquo.
 - Permite que você se mantenha sensível e ágil o suficiente para reconhecer mudanças e se planejar para solucionar problemas.
 - Permite que as lições aprendidas nos seus experimentos possam servir de apoio a outros projetos (seus ou não)
- Assim, o primeiro passo é definir o contexto do seu sistema e os seus objetivos e tudo isso que falamos pode ajudar.

Teste Suas Ideias

- Antes de meter a mão na massa e depois de definidos os objetivos teste suas ideias.
- Qual é o objetivo de testar ideias? Refinar as ideias!!
- Como? Exemplos:
 - Construir protótipos de baixa fidelidade (papel, quadro, storyboards) mas que permitam a discussão (teste) de ideias.
 - Usar vídeos para discutir as ideias. Apesar de mais custosos para produzir podem ser úteis para explicar a origem do sistema no futuro.
 - Criar protótipos rápidos da UI podem permitir simular um ambiente mais real e obter dados mais concisos sobre as reações de um usuário.
 - Usar a técnica do “Mágico de OZ”

Teste Suas Ideias

- Testes em laboratórios são chatos e muito pouco proveitosos se não existe a participação dos "ratos"!!.
- Uma abordagem mais simples pode ser: Apresentar o cenário a uma pessoa (de fora da equipe) e colher a impressão.
- Em casos de críticas avaliar e modificar o cenário.
- Este ciclo pode se repetir várias vezes até que um cenário estável seja alcançado.
- Custa muito menos que construir (programar, soldar, etc.) e pode trazer resultados muito melhores.

Design de Sistemas

- Uma vez que você está feliz com suas ideias e que seus objetivos foram definidos (não necessariamente nesta ordem!)
- É preciso desenhar, projetar ou modelar seu sistema para "vestir" os objetivos.
- Para isso algumas preocupações que são específicas da Ubicomp precisam ser entendidas.

Conhecimento Computacional do Mundo Físico

- Na perspectiva do design de sistemas, não é claro quais interfaces ou mesmo parte internas (sistemas ubicomp) precisam de uma abordagem experimental.
- Novamente, como Weiser disse:
 - Estes sistemas devem ser invisíveis, calmos e estar focados na resolução de atividades diárias dos usuários.
- Para isso é importante saber os limites entre o sistema e os seus usuários. ("Tratabilidade" computacional e suporte ao usuário)
- Isso significa: Entender as barreiras entre o mundo físico e o mundo virtual.

Conhecimento Computacional do Mundo Físico

- Na computação ubíqua:
 - Muitas Interfaces
 - Envolvem muitos dispositivos
 - Diferentes Inputs e Outputs.
 - Sensores embarcados ou do ambiente
 - Incorporações de dispositivos/usuários de forma sutil
 - formas oblíquas (não convencional) de interação.
- Semântica “rubicom” (Fox and Kindberg 2002)
 - Termo que encapsula a divisão entre a responsabilidade do usuário e do sistema.
 - Demarca quais as responsabilidades das decisões tomadas entre o sistema e os usuários.

Conhecimento Computacional do Mundo Físico

- Semântica “rubicom” em termos de design de sistemas:
 - Implica em definir qual o conhecimento que o sistema tem (pode ter) do mundo físico e do comportamento dos usuários (através de sensores e da interação com usuários).
 - Contrapartida: O conhecimento que o usuário tem do sistema e como ele pode influencia-lo.
 - Definir mecanismos e permissões de interação de maneira que um possa influenciar o outro.

Conhecimento Computacional do Mundo Físico

- Como um designer de sistemas você deve:
 - Decidir qual conhecimento (sobre algum funcionamento do mundo real) o seu sistema deve possuir.
 - Como esse conhecimento vai ser obtido pelo sistema.
 - Como esse conhecimento vai ser representado.
 - Como manter/alterar o estado do sistema.
- A não ser que essas informações sejam simples de medir (sentir) é preciso decidir o que fazer.
- Se elas forem imperfeitas ou se as conclusões a partir delas forem imprecisas ?

Conhecimento Computacional do Mundo Físico

- A aceitabilidade do sistema tem como ponto crucial o design envolvendo as decisões do usuário, do sistema ou ainda rubicom (onde a decisão de um pode ser a do outro).
- Três questões chaves devem ser respondidas:
 - O que pode ser sentido (Sensor) de forma confiável?
 - O que pode ser conhecido de forma confiável?
 - O que pode ser inferido (deduzido) de forma confiável?
- O grau de confiabilidade nas suas respostas para essas questões quando confrontados com funções do seu sistema pode determinar se elas são possíveis ou se você encontrou um desafio de pesquisa!

Descontinuidade, Sensibilidade e Ignorância Tolerada

- Existem limites claros entre o que o seu sistema pode saber sobre o mundo real e o sobre as pessoas que o habitam.
- Cada tipo de sensor tem propriedades inatas e características físicas que “governam” o que e quanto pode ser medido.

Descontinuidade, Sensibilidade e Ignorância Tolerada

- Sensores podem não ter potência suficiente para cobrir uma área ou detectar uma atividade.
- Eles podem não estar posicionados corretamente (densidade).
- O Sinal do GPS por exemplo varia de acordo com o local do receptor. Sombras podem prejudicar o sinal

Descontinuidade, Sensibilidade e Ignorância Tolerada

- A atividade que você quer detectar é complexa devido as sutilezas do mundo real.
- Outras atividades podem interferir (gerando barulhos, temperatura, vento) e assim fica difícil de isolar apenas uma atividade.
- Existe ainda uma questão de confiança sobre o que fazer no caso de uma falha (parcial ou total) de um sensor.

Descontinuidade, Sensibilidade e Ignorância Tolerada

- A Ignorância Tolerada (IT - Friday et al., 2005).
 - Como exibir (aceitar) a Ignorância?
 - Como projetar (design) esses sistemas?
- Embora seja correto projetar sistemas que mantenham o máximo de funções quando falhas ocorrerem, o primeiro passo é definir o escopo e limites do seu sistemas.

Descontinuidade, Sensibilidade e Ignorância Tolerada

- O termo “seamful design” (design descontinuado) foi apresentado por Chalmers et al. (2003).
 - Usa as emendas (costuras) ou os limites de um sistema como fonte para os designers do sistema.
 - Ex.: Jogo “Can you see me now?”
 - Corredores de rua (reais) têm que alcançar corredores virtuais (jogadores virtuais). Ambos estão sobreposto no mesmo espaço (um real e outro virtual).

Descontinuidade, Sensibilidade e Ignorância Tolerada

- Ex.: Jogo "Can you see me now?"
 - Corredores de rua rastreados por GPS.
 - Ficou aparente uma "costura" do sistema ... Capacidade de rastrear corredores.
 - Análise do log do jogo: erros de 4 a 106 metros no GPS. Áreas abertas erro menor, áreas com prédios altos erro maior.

Descontinuidade, Sensibilidade e Ignorância Tolerada

- Ex.: Jogo "Can you see me now?"
 - Depois de um tempo os corredores de rua perceberam essa "emenda" e começaram a traçar táticas para ganhar o jogo. Escolher rotas onde espaços abertos eram mais frequentes!! (preocupação com o GPS e com os corredores virtuais)

Descontinuidade, Sensibilidade e Ignorância Tolerada



Descontinuidade, Sensibilidade e Ignorância Tolerada

- No caso do jogo às limitações permitiram aos corredores melhorar a sua habilidade de jogar.
- Uma questão relacionada ao design é: quanto das "emendas" de um sistema devem ser expostas?

Descontinuidade, Sensibilidade e Ignorância Tolerada

- Quatro estratégias (Chalmers et al. (2003)):
 - Pessimista: Apenas apresentar informações seguramente corretas.
 - Otimista: Mostrar tudo como se fosse correto.
 - Cautelosa: Expor as incertezas explicitamente.
 - Oportunista: Explorar as incertezas (ver em Gaver et al., 2003).

Descontinuidade, Sensibilidade e Ignorância Tolerada

- Alguns podem considerar as estratégias pessimista e oportunista como mais tradicionais.
 - Ex.: é muito comum representar a posição de um usuário em um mapa como um ponto (metáfora).
 - O ponto (apenas ele) não é adequado para transmitir "incerteza", "imprecisão" de localização.
 - O ponto não reflete se a localização é algo que o sistema acredita ser correto. (Em uma área de sombra o ponto pode ser o ultimo capturado pelo receptor.)

Descontinuidade, Sensibilidade e Ignorância Tolerada

- O usuário de um sistema de mapas pode aproximar/afastar a imagem e assim ganhar confiança no sistema (mesmo com informações imprecisas).
- A abordagem cautelosa é usada por exemplo em celulares: Barra de sinal.
- Reflete a possibilidade de sucesso em executar uma chamada.
- Com as abordagens cautelosas e oportunistas usuários são mais receptivos.

Descontinuidade, Sensibilidade e Ignorância Tolerada

- O que não pode ser sentido (sensores) ou o quanto pode ou dever ser exibido de “emendas de um sistema” são questões que devem ser usadas como fonte para um bom design Ubicomp.
- Três palavras importantes: Sensato, Possível e Desejável (ver Boucher et al., 2003)
 - Taxonomia que ajuda a classificar dispositivos e suas interações de modo a exibir oportunidades não conhecidas a priori.

Modelo Mental de Usuário e Responsabilidade

- O corolário (consequência direta) da descontinuidade de um sistema e da semântica rubicom é planejar qual o papel do usuário na operação de um sistema.
- Sistemas Ubicom diferem no grau de inteligência e entendimento que querem mostrar aos usuários no sentido que satisfazer seus objetivos e desejos.
- Existem um espectro de escolhas de design em Ubicomp para decidir quando envolver um usuário no sensoriamento/entendimento do mundo físico e na tomada (ou pelo menos instigá-lo) de decisões.

Modelo Mental de Usuário e Responsabilidade

- Um cenário poderia ser: O sistema entende completamente o que o usuário quer e toma decisões preemptivas ao usuário.
- Outro cenário (mais cauteloso): O sistema não toma nenhuma decisão sem a anuência do usuário. Ou ainda, ele é quem deve fornecer a informação de sensoriamento.
- Um posicionamento: deve-se automatizar funcionalidades de maneira a atender as necessidades de um usuário....MAS

Modelo Mental de Usuário e Responsabilidade

- É preciso dar algum poder de intervenção para usuários.
- Outro cenário seria aprender com as ações dos usuários e tomar decisões automaticamente (ou supervisionadas). Voltar a aprender e novamente tomar decisões (WAZE faz isso com as rotas...)

Modelo Mental de Usuário e Responsabilidade

- Para ajudar, considere :
 - A inconveniência ou frequência do envolvimento do usuário;
 - A severidade ou inconveniência se algo der errado no sistema;
 - A confiabilidade de achar o momento apropriado para a ação apropriada;
 - A aceitabilidade do usuário para ações automatizadas.
- O Waze usa a interferência do usuário para se atualizar!!!! (Acidentes, novas estradas ainda não mapeadas, Radar...etc).

Modelo Mental de Usuário e Responsabilidade

- A pergunta chave é: O que você pretende que o usuário perceba como o "sistema em operação"?
- Para ficar confortável com o sistema, adotá-lo e se apropriar do sistema o usuário deve:
 - Estar apto a formar um modelo mental de causa e efeito ou com uma lógica plausível do seu funcionamento.
- Em sistemas ubíquos mais artísticos esta questão é mais provocativa...mas ainda assim é algo que deve ser levado em consideração no design.

Tudo é sempre em tempo de execução

- Sistemas Ubíquos são: Heterogêneos, voláteis, flutuantes, as partes não são plenamente confiáveis, as conexões não são contínuas....Mas....
- Uma vez implantados TUDO acontece em tempo de execução.
- Dificilmente você terá acesso de forma simultânea a todas as partes (para boot ou upgrade por exemplo!!).

Tudo é sempre em tempo de execução

- Algumas implicações:
 - Sistemas que requerem uma ordem de inicialização tendem a falhar.
 - Se a disponibilidade de elementos é esporádica o seu sistema deve estar apto a (sutilmente) gerenciar desconexões, novas conexões, reconexões, etc.
 - Assuma que partes dos seus sistema podem falhar (ou estar temporariamente fora do ar/alcançe). Projeto o seu sistema para suportar isso e possibilitar recuperação do estado.

Tudo é sempre em tempo de execução

- Algumas implicações:
 - Decida de forma proativa como manipular dados em caso de desconexão e posterior reconexão (buffer, banco, disco, etc.). Estabeleça limites para isso (quanto tempo, quantos dados, etc.).
 - Considere usar protocolos de controle de versão para poder identificar erros relacionados a versões anteriores.

Manipulação de Conexões Transientes

- A rede (ou a falta e falha dela) tem implicações sérias em sistemas ubíquos.
 - Resolução de nomes para de funcionar;
 - Exceções em softwares pelo fechamento inesperado de conexões podem levar ao congelamento de interfaces de usuário ou diminuição do número de funcionalidades.
 - Considerar o que fazer quando partes do seu sistema, que deveriam estar sempre disponíveis, falharem ajuda a identificar problemas e soluções previamente.

Manipulação de Conexões Transientes

- Quando a rede falha é comum armazenar dados em buffers (em muitos níveis das camadas de protocolos).
- TCP faz isso identificando cada porção de dado.
- Quando dados (em tempo real) são necessários para interação do usuário é comum (quando uma falha acontece) que ele tente interagir muitas vezes (o que piora o problema!!!).
- Por isso é importante levar esses aspectos em consideração.

Manter o Estado

- Falhas na rede e nos dispositivos têm impacto direto em sistemas Ubicomp.
- É importante definir estratégias para tratar nos dois casos.
- Partes do seu sistema podem se adequar a técnicas bem conhecidas para mascarar problemas e suportar em algum grau de tolerância.
- Mas na maioria dos sistemas ubíquos isso não é verdade.

Manter o estado

- Eles podem estar intimamente ligados a um determinado hardware. Podem estar em locais específicos.
- Isso pode invaibilizar o uso de técnicas tradicionais como replicação, redundância, etc. Nesses casos é preciso alguma alternativa.
- Na literatura existem algumas técnicas reportadas e entre elas podem ser destacas:

Manter o estado

- Replicação Otimista de Estado: Permite que partes desconectadas de um sistema funcionem e se atualizem depois de uma reconexão.
- Convergência em Estado Consistente: Quando usamos redes sociais para combinar as possíveis datas de um encontro. Cada usuário sugere ou escolhe datas... depois de um tempo existe uma tendência a convergir para uma data e horário (estado consistente).

Manter o estado

- Uso de armazenamento persistente (local ou remoto)
- Exteriorizar o estado do sistema através de middleware e dessa forma permitir que partes do sistema possam recuperar / atualizar o estado.
- Uso de cache em *peers* abrindo a possibilidade de recuperação no futuro (em caso de necessidade)
- Uso de protocolos de propagação epidêmica de estado (FOFOCA - Gossip protocols)
- CUIDADO com o tamanho da replicação: Os recursos são limitados (normalmente).

Funciona?

- Como "debugar" um sistema Ubíquo? Desafiador!!!
- Muitos elementos, diferentes e muitas vezes inacessíveis remotamente.
- Alguns elementos não tem interface com usuário.
- O que fazer?
 - Monitorar as "saídas" do sistema.
 - Quando for possível, "injetar" mensagens de teste para checar reações do sistema.
 - uso de logs, análise de pacotes de rede, uso de protocolos de status, uso de telas de status (ex.: LEDs), interfaces de diagnóstico (como webservers), sheells,

E depois?

- Escolha bem as partes do seu sistema. Saia da concha! Leve em consideração a integração dos componentes.
- Implantar em casa antes de colocar o bloco na rua. Comer sua própria comida antes de vender quentinha.
- Espere o inesperado!

Outra Boa Referência

