Aplicações Móveis

Exercícios de Revisão

Coordenação de Engenharia Informática
Departamento de Engenharias e Tecnologias
Instituto Superior Politécnico de Tecnologias e Ciências

Nome : Marcelo Rocha - 20210032

Sensibilidade ao Contexto

1. **Qual é a definição de contexto? Forneça um exemplo que ilustre a relevância da informação.**

   R: Contexto é qualquer informação que pode ser utilizada para caracterizar a situação de uma entidade, seja uma pessoa, local ou objeto, que é relevante para a interação entre o usuário e uma aplicação, conforme a definição de Dey. O contexto vai além da simples localização e pode envolver identidade, atividade e estado, ampliando as possibilidades de adaptação de uma aplicação conforme o ambiente ou situação em que o usuário se encontra.

   Exemplo: Em um aplicativo de um museu, o contexto relevante para ajudar os visitantes pode ser sua localização e idioma. Essas informações são úteis para guiar a visita e oferecer conteúdo adequado. Entretanto, outras informações, como temperatura corporal ou estado civil dos visitantes, não agregam à experiência e são descartadas.

2. **O que é um sistema sensível ao contexto?**

   R: Um sistema sensível ao contexto é aquele que utiliza o contexto para oferecer informações e/ou serviços relevantes ao usuário, conforme as necessidades de sua tarefa (relevância dependente da tarefa do utilizador).

3. **Existem três características principais que os sistemas sensíveis ao contexto podem oferecer. Quais são essas?**

   R: **Apresentação de informações e serviços ao usuário:** o sistema fornece informações ao usuário, enriquecidas com dados contextuais, ou disponibiliza serviços baseados no contexto atual do usuário (por exemplo, uma lista de contatos com a localização ou sugestões de restaurantes próximos).

   Execução automática de um serviço: o sistema executa um serviço automaticamente com base no contexto atual (por exemplo, atualizar o status em uma rede social com base em dados de movimento, como "dormindo" ou "caminhando").

Associação de contexto a informações para recuperação posterior: o sistema permite associar dados digitais ao contexto do usuário, possibilitando sua recuperação no futuro (por exemplo, notas virtuais anexadas a locais específicos, visíveis para outros usuários).

4. **Em relação à categorização das opções arquitecturais disponíveis para o desenvolvedor de aplicações de sistemas de contexto distribuído, existem quatro camadas: captura, inferência, distribuição e consumo. Para a camada de captura e inferência, forneça uma breve descrição da funcionalidade de cada camada; também apresente uma figura que ilustra como as quatro camadas são dispostas em camadas.**

R: **Camada de Captura (Layer 0 - Context sensors and actuators):** Esta camada envolve sensores que capturam dados do ambiente e atuadores que podem modificar o ambiente. Eles são responsáveis por coletar o contexto, como localização, temperatura, etc.

Camada de Inferência (Layer 1 - Context processing components): Nessa camada, os dados brutos coletados pelos sensores são processados e transformados em informações de contexto utilizáveis. Isso envolve interpretar os dados para identificar condições relevantes para o sistema.

Camada de Distribuição (Layer 2 - Context repositories): Aqui, o contexto é armazenado e distribuído. A camada de repositórios de contexto permite o armazenamento persistente dos dados de contexto e oferece mecanismos de consulta para fornecer os dados relevantes quando necessários.

Camada de Consumo (Layer 3 e Layer 4 - Decision support tools e Application components): Nessa camada, os componentes de suporte à decisão e as aplicações de usuário final consomem as informações de contexto para adaptar seu comportamento. As ferramentas de suporte à decisão ajudam a identificar a melhor ação a ser tomada com base no contexto disponível, e os componentes de aplicação interagem diretamente com o usuário.

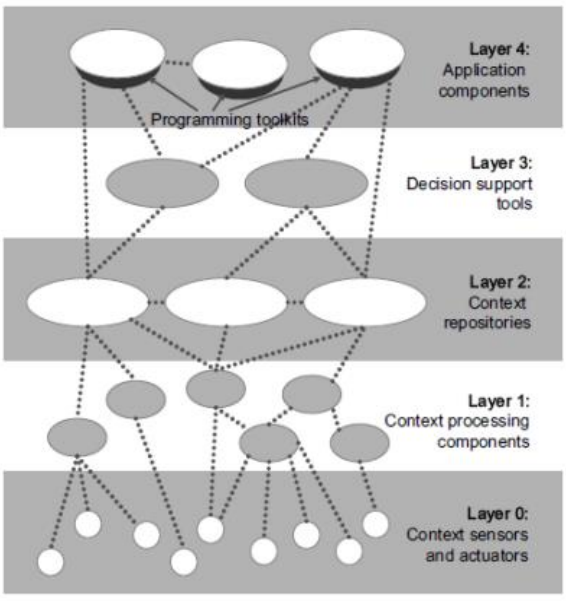

Figure 2: Layers of a context-aware application according to Henricksen

5. **Em sistemas sensíveis ao contexto, a palavra sensor não se refere apenas ao hardware de detecção, mas também a todas as fontes de dados que podem fornecer informações de contexto utilizáveis, levando assim a sensores físicos e virtuais. Concorda (responda "sim" ou "não")? Forneça: i) a definição de cada tipo de sensor (físico e virtual), e ii) um exemplo de sensor físico e outro de sensor virtual.**

R: Sim, concordo que a palavra "sensor" em sistemas sensíveis ao contexto não se refere apenas a hardware, mas a todas as fontes de dados que fornecem informações de contexto utilizáveis.

Sensor Físico: São dispositivos de hardware que capturam dados físicos, como luz, som, movimento e localização no ambiente. Eles detectam diretamente o ambiente físico e enviam dados em tempo real. Um exemplo de sensor físico seria um GPS (Global Positioning System), que fornece as coordenadas de localização de um dispositivo no ambiente físico.

Sensor Virtual: São fontes de dados que obtêm informações a partir de software, sistemas operacionais, redes ou aplicações. Eles capturam informações sem a necessidade de hardware físico adicional, muitas vezes utilizando infraestrutura já existente em sistemas digitais. Um exemplo de sensor virtual seria um sistema de calendário em rede, que detecta eventos ou compromissos e pode fornecer informações sobre a atividade e localização do usuário com base nos dados do software.

6. **Considere a seguinte frase: "Para que a proactividade seja eficaz, é crucial que um sistema de computação pervasiva rastreie a intenção do utilizador." Concorda com esta frase? A intenção do utilizador é a única informação relevante para um sistema ser proactivo? Justifique sua resposta.**

R: Não, a intenção do utilizador não é a única informação relevante para que um sistema de computação pervasiva seja proactivo. Embora a intenção do utilizador seja uma peça importante para a proactividade, existem outras informações cruciais que também precisam ser consideradas para que o sistema seja eficaz:

- **Contexto :** A proactividade eficaz também depende de entender o contexto em que o utilizador está inserido, isso inclui informações como: **Localização**: saber onde o utilizador está pode ajudar o sistema a fornecer informações ou ações relevantes, como sugerir rotas ou ajustar preferências de iluminação ou som. **Identidade**: O perfil do utilizador pode influenciar as sugestões e ações automáticas, adaptando-se a diferentes preferências ou necessidades individuais. **Atividade**: Compreender se o utilizador está, por exemplo, dirigindo, caminhando ou descansando pode ajudar a determinar o momento adequado para a intervenção do sistema. **Tempo**: O horário pode determinar que tipo de informações ou funcionalidades são úteis ou irrelevantes naquele momento.

- **Preferências Pessoais:** Mesmo que o sistema entenda a intenção do utilizador, também é importante levar em consideração as suas preferências pessoais e os seus hábitos. Nem todas as intenções são claras ou explícitas, então um sistema proactivo deve aprender padrões de comportamento ao longo do tempo.

- **Dados Históricos e Padrões:** Um sistema de computação pervasiva pode usar dados históricos e padrões de uso para prever necessidades futuras. Por exemplo, se um utilizador geralmente faz uma determinada tarefa em um horário específico, o sistema pode sugerir ou automatizar essa ação com base nesse padrão.

- **Feedback do Utilizador:** A proactividade deve ser refinada continuamente com base no feedback do utilizador. Um sistema que apenas rastreia intenções, mas não se adapta ao feedback, pode acabar sendo ineficaz ou até irritante.

7. **Considerar a necessidade de um sistema móvel/ubíquo se adaptar automaticamente ao contexto. Apresente um exemplo em que a alta velocidade na qual um sistema é capaz de se adaptar é boa e outro exemplo em que essa alta velocidade não é boa.**

R: A alta velocidade de adaptação em sistemas móveis/ubíquos pode ser tanto benéfica quanto prejudicial, dependendo do contexto.

Um exemplo onde alta velocidade é boa seria a navegação por GPS. Em um sistema de navegação, a alta velocidade de adaptação é crucial para recalcular rotas rapidamente ao detectar mudanças de direção ou imprevistos, como trânsito ou desvios. Isso permite que o usuário tenha direções atualizadas em tempo real, o que é essencial para uma navegação eficaz.

Um exemplo onde alta velocidade não é boa seria a mudança automática de brilho da tela. Se um sistema ajustar o brilho da tela rapidamente de acordo com mudanças mínimas de luz ambiente, isso pode resultar em uma experiência incômoda, com mudanças constantes e abruptas de brilho que distrai o usuário. Um tempo de resposta mais lento pode proporcionar uma experiência mais confortável e fluida.

8. **Além do grande número de utilizadores e mensagens trocadas, a propagação de contexto cria desafios únicos no domínio dos sistemas distribuídos. Que desafios são esses? Forneça exemplos que ilustrem cada desafio.**

R: A propagação de contexto em sistemas distribuídos apresenta vários desafios únicos, que podem ser agrupados em algumas categorias principais:

- **Escalabilidade:** À medida que o número de usuários e dispositivos aumenta, a quantidade de dados contextuais a ser gerenciada e transmitida também cresce. Isso pode levar a problemas de desempenho e latência. Exemplo: Em um evento com milhares de participantes, um sistema que tenta enviar atualizações de contexto (como localização ou status) para todos os usuários pode enfrentar congestionamento na rede, resultando em atrasos na entrega das informações.

- **Dinamicidade:** Os ambientes em que os sistemas operam são frequentemente dinâmicos, com usuários e dispositivos mudando de localização e estado rapidamente. Isso torna difícil manter informações de contexto atualizadas e precisas. Exemplo: Em um ambiente de escritório, um usuário pode se mover

entre salas de reunião, e o sistema precisa atualizar rapidamente sua localização para fornecer informações relevantes, como a disponibilidade de salas ou colegas.

- **Heterogeneidade:** Os sistemas distribuídos podem incluir uma variedade de dispositivos e plataformas, cada um com diferentes capacidades e formatos de dados. Isso pode complicar a integração e a interpretação dos dados contextuais.Exemplo: Um sistema que coleta dados de sensores de diferentes fabricantes (como sensores de temperatura, umidade e movimento) pode enfrentar dificuldades em unificar esses dados em um formato comum para análise.

- **Privacidade e Segurança:** A coleta e a propagação de dados contextuais levantam preocupações sobre a privacidade dos usuários e a segurança das informações. É crucial garantir que os dados sejam transmitidos de forma segura e que os usuários tenham controle sobre quais informações são compartilhadas.Exemplo: Um aplicativo de localização que compartilha a posição do usuário com amigos pode ser vulnerável a abusos se não houver controles adequados de privacidade, permitindo que terceiros acessem informações sensíveis sem consentimento.

- **Confiabilidade:** A propagação de contexto deve ser confiável, garantindo que as informações sejam entregues corretamente e em tempo hábil. Falhas na comunicação podem resultar em decisões baseadas em dados desatualizados ou incorretos. Exemplo: Em um sistema de emergência, se a localização de um usuário em perigo não for atualizada corretamente, os serviços de emergência podem ser enviados para o local errado, comprometendo a eficácia da resposta.

9. **Onde a camada de inferência de contexto pode ser executada? A localização está intimamente relacionada com várias propriedades de um sistema sensível ao contexto; quais são esses?**

R: A camada de inferência de contexto, também conhecida como camada de pré-processamento, pode ser executada em diferentes localizações dentro de um sistema sensível ao contexto: nos sensores, em um componente middleware (servidor) ou na aplicação final. A localização da camada de inferência é importante porque influencia várias propriedades do sistema, incluindo o consumo de largura de banda da rede, a complexidade em termos de consumo de CPU/memória, a reutilização da inferência e a personalização.

- **Consumo de Largura de Banda da Rede:** Quando a inferência de contexto é realizada próximo ao sensor, como em dispositivos com capacidade de processar informações em tempo real, a quantidade de dados transmitidos pela rede é reduzida. Isso é especialmente relevante em sistemas distribuídos com dispositivos dispersos, onde a largura de banda pode ser limitada ou as conexões podem ser pouco confiáveis. Por exemplo, em um sistema de monitoramento de saúde, se um sensor de atividade física pode transformar dados brutos em uma classificação simples (como "ativo" ou "inativo") antes de enviar, isso diminui o volume de dados transmitidos.

- **Complexidade (Consumo de CPU/Memória):** A execução da inferência em dispositivos de capacidade limitada, como smartphones, pode resultar em alto consumo de CPU e memória, especialmente ao aplicar algoritmos sofisticados, como Redes Neurais. Para evitar isso, é comum mover a inferência para um servidor mais poderoso, onde os recursos são mais abundantes. Por exemplo, um aplicativo que detecta atividades do usuário pode coletar dados brutos de um acelerômetro e enviá-los para um servidor, que então realiza a inferência mais complexa para determinar a atividade (como correr ou caminhar).

- **Reutilização:** A reutilização se refere à capacidade de usar a mesma inferência de contexto em diferentes aplicações. Quando a inferência é realizada perto do sensor, é possível criar dados de contexto de alto nível que podem ser utilizados por várias aplicações. Por exemplo, a transformação de coordenadas GPS em nomes de ruas é uma tarefa que pode ser reutilizada em diversos aplicativos de localização, evitando que cada um implemente sua própria lógica.

- **Personalização:** A personalização permite que a inferência seja ajustada para atender às necessidades específicas dos usuários. Por exemplo, um aplicativo que permite aos usuários definir suas próprias atividades ou interpretar movimentos específicos de forma personalizada pode realizar a inferência na aplicação. No entanto, isso pode levar a problemas de escalabilidade, pois a personalização para cada usuário pode complicar a implementação. Sistemas que realizam a inferência em middleware podem encontrar um equilíbrio entre personalização e reutilização, adaptando-se às necessidades dos usuários sem sobrecarregar as aplicações individuais.

10. **O contexto deve ser distribuído entre os componentes do sistema (sensores, *middleware*, aplicação). Quais abordagens podem ser usadas? Quais são os principais aspectos distintivos de cada um?**

R: A distribuição de contexto entre os componentes de um sistema sensível ao contexto, como sensores, middleware e aplicações, pode ser realizada através de várias abordagens. Cada uma delas possui características distintas que impactam na eficiência e na eficácia do sistema. Aqui estão algumas das principais abordagens e seus aspectos distintivos:

- **Arquitetura Centralizada** : Neste modelo, um middleware centralizado serve como intermediário entre os sensores dispersos e as aplicações cliente. Exemplo: o sistema CenceMe, que usa um servidor HTTP para compartilhar contexto pessoal.
    - **Aspectos Distintivos:**
        - **Simplicidade de Comunicação :** Os dispositivos se comunicam apenas com um único servidor, facilitando a configuração .

- **Alívio da Carga em Dispositivos:** Alivia dispositivos com recursos limitados de tarefas exigentes em CPU e memória.

- **Ponto Único de Falha:** Se o servidor falhar, todo o sistema pode ser comprometido.

- **Escalabilidade Limitada:** O desempenho pode diminuir à medida que mais dispositivos se conectam.

- **Arquitetura Partitionada ou Federada:** Aqui, múltiplos servidores são utilizados, onde cada dispositivo se conecta a um servidor específico, e os servidores se comunicam entre si para compartilhar informações. Este modelo é chamado de arquitetura em acíclica peer-to-peer.

  - **Aspectos Distintivos:**

    - **Distribuição da Carga:** Melhora a escalabilidade ao distribuir a carga entre vários servidores.

    - **Desafio na Divisão de Usuários:** A segmentação de usuários pode ser complexa, especialmente se os identificadores não forem uniformes.

    - **Aumento de Atraso na Transmissão:** A comunicação entre servidores pode introduzir atrasos adicionais.

- **Arquitetura Peer-to-Peer (P2P):** Nesta abordagem, cada dispositivo comunica diretamente com outros dispositivos, sem a necessidade de um servidor central. O middleware está incorporado em cada dispositivo.

  - **Aspectos Distintivos:**

    - **Resiliência:** Não há ponto único de falha, pois cada dispositivo pode atuar como cliente e servidor.

    - **Robustez a Problemas de Rede:** Múltiplos caminhos de comunicação tornam o sistema mais resiliente a falhas de rede.
    - **Complexidade na Entrega de Contexto:** Algoritmos complexos são necessários para garantir que as informações de contexto sejam compartilhadas eficientemente.

- **Abordagens Híbridas:** Combina diferentes modelos de distribuição, como a arquitetura AwareMedia, que usa uma mistura de modelos centralizados e peer-to-peer.

    - **Aspectos Distintivos:**

        - **Flexibilidade:** Permite que dispositivos se comuniquem com servidores e entre si, aproveitando os benefícios de ambos os modelos.

        - **Melhor Gerenciamento de Recursos:** Combina as forças de várias abordagens para otimizar a distribuição de contexto.

11. **Em relação à camada de consumo, quais são as várias opções e como funcionam?**

R: A camada de consumo em sistemas contextuais permite que aplicativos cliente obtenham informações de contexto. Existem duas opções principais para essa interação: Push e Pull.

Nos sistemas baseados em Pull, o consumidor (aplicação cliente) consulta o componente que possui a informação (sensor ou middleware) para obter atualizações. Isso pode ser feito manualmente (por exemplo, o usuário clica no botão "atualizar") ou periodicamente (por exemplo, verificando atualizações a cada 5 minutos). Nos sistemas baseados em Push, o componente que possui a informação é responsável por entregá-la às aplicações cliente interessadas. A principal distinção entre essas abordagens reside em quem inicia a comunicação. Se uma aplicação cliente inicia a comunicação, o sistema é baseado em Pull; caso contrário, é baseado em Push.

Os mecanismos baseados em Pull são simples de implementar, pois a aplicação cliente apenas precisa consultar um determinado componente com uma certa periodicidade. No cenário extremo, a aplicação pode simplesmente delegar a responsabilidade de atualizar a informação ao usuário. Além disso, essa periodicidade (intervalo de polling) pode mudar de acordo com as necessidades atuais da aplicação. Por exemplo, uma aplicação que roda em um celular pode aumentar o intervalo de polling para economizar a vida útil da bateria, quando a energia da bateria cai abaixo de um certo limite. No entanto, esse mecanismo sofre de dois problemas que podem ser críticos em um sistema consciente do contexto. Primeiro, a informação de contexto chega à aplicação com um atraso. No pior cenário, esse atraso é igual ao intervalo de polling. Se o intervalo de polling é definido como 5 minutos, haverá casos em que a aplicação recebe informações de contexto de 5 minutos atrás, comprometendo sua utilidade. A aplicação pode minimizar essa desvantagem reduzindo o intervalo de polling, mas então o segundo problema pode começar a ocorrer: a maioria das consultas será desnecessária, uma vez que não há novas informações de contexto disponíveis. Como a aplicação não sabe quando haverá atualizações, não tem alternativa a não ser continuar perguntando até que haja nova informação, levando ao desperdício de largura de banda e consumo de CPU.

Os mecanismos baseados em Push são mais complexos do que os mecanismos baseados em Pull. Como o componente que possui a informação também é responsável

por entregá-la, esse componente precisa saber como alcançar todos os possíveis consumidores interessados nessa informação. Geralmente, é necessária uma conexão permanente com todos os consumidores. Se o sistema tiver um grande número de consumidores, seu desempenho pode degradar. Além disso, esses sistemas têm uma escalabilidade ruim, uma vez que cada novo consumidor degrada ainda mais o desempenho. Existem duas medidas para superar o problema de escalabilidade: reduzir o escopo e/ou relaxar o tempo de entrega.