Implementarea de scenarii AmI în AmILab

Cristian Neagoe, Cristian Grigoraș, Vlad Herescu

Coordonator: ș.l. Andrei Olaru

Inteligența ambientală

Subiectul stagiului

Dispozitivele Android facilitează implementarea unui sistem de Inteligență Ambientală

- **Calcul omniprezent (*ubiquitous computing)***
- **Context-Awareness**
- **Adaptabilitate**
- **Proactivitate**
- **Profiling**
- **Facilitează comunicația dintre oameni, activitatea utilizatorului și interacțiunea acestuia cu tehnologia**

Scenariu

Elemente de scenariu AmI

Din analiza diverselor scenarii am extras următoarele:

- determinarea locației pe baza informațiilor despre rețelele wireless
- schimbarea volumului apelului în funcție de zgomotul din sală
- determinarea starea de mișcare a utlizatorului pe baza datelor accelerometrului
- monitorizarea schimbărilor asupra fișierelor de lucru
- sistem de notificări pro-active
- transfer automat de fișiere între dispozitive

Instrumente colaborative

- Eclipse & Plugins
  - EGit: lucrul cu GitHub
  - ADT: dezvoltare software Android
  - Copyright Wizard: aplicare copyright pe surse
  - ObjectAid: realizare diagrame UML
  - JDK 6: versiunea necesară de Java pentru dezvoltare pe Android

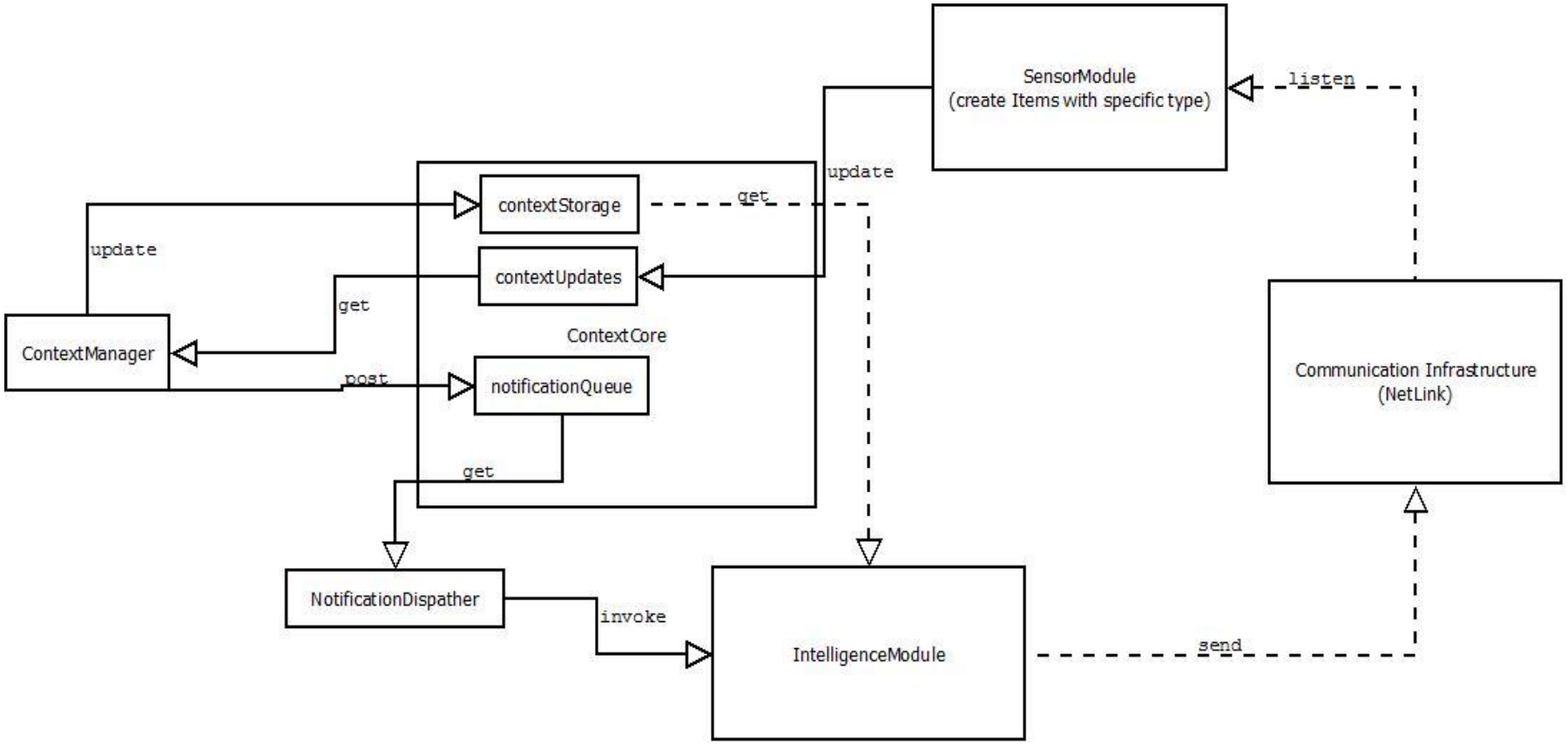
Arhitectură

Componente

- ContextCore
  - contextUpdates, notificationQueue, contextStorage
- Managers:
  - ContextManager, NotificationDispatcher
- Module:
  - implementări de SensorModule și IntelligenceModule
- Tipuri de date:
  - ContextItem, ContextTypes

Arhitectură

Detalii funcționare



Context-awareness

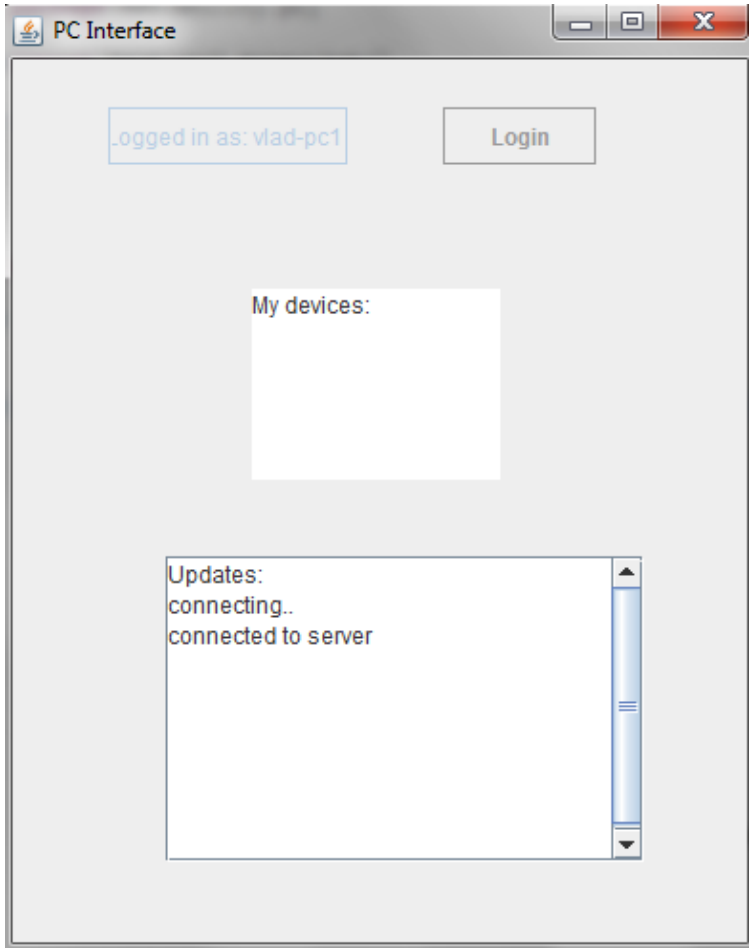
Tipuri de context

- Inteligența ambientală se bazează pe transformări succesive ale unui context inițial
- Contextul = starea individului şi a mediului înconjurător
- Tipuri de context:
  - Persoana se mişcă sau stă pe loc *(Accelerometer Context)*
  - Persoana lucrează la un fişier *(File Context)*
  - În încăpere e zgomot sau nu *(Sound Context)*
  - Locația în care te afli *(Location Context)*

Implementare

Notificări & Interfețe

- PC:
  - login
  - notificări sunet/mișcare
  - listă cu dispozitivele mele
- Android:
  - login
  - notificări sunet/mișcare
  - listă cu dispozitivele mele
- Percepții (valorile senzorilor)
  - sunet
  - mișcare
- Help Q&A:
  - cerere ajutor altui peer
  - răspunsul acestuia



Implementare

Servicii și Activități Android

- LoginActivity: fereastra de autentificare pe Android
  - username
  - buton login
- MainActivity: fereastra principală
  - zonă de text pentru lista dispozitivelor mele
  - zonă de text pentru notificări (conectare, percepții, copiere fișiere)
  - alertDialog pentru cerere de ajutor
- Servicii senzor :
  - sunet
  - accelerometru
  - wireless networks

Implementare

NetLink & Transfer

- Scop:
  - Comunicarea între dispozitive
- Componente NetLink:
  - Interacțiunea client – server (persistență):
    - Create connection with server, receive from server (hello & Connections)
  - Interacțiunea client – client (send and close):
    - Send, receive
  - Interacțiunea server – client:
    - Listen permanent şi asocierea deviceurilor pe baza id-ului
- Funcționare și utilitate:
  - Pe socket
  - Transferul de items (fișiere, percepții)

```
ObjectOutputStream out = new ObjectOutputStream(socket.getOutputStream);

Out.writeObject(Object o);
```

Implementare

Accelerometru

- Scop:
  - Deciderea stării de mişcare a individului
- Componente:
  - `mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);`
  - `ImAccelerometer = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);`
  - `mSensorManager.registerListener(this, mAccelerometer,SensorManager.SENSOR_DELAY_NORMAL);`
- Funcționare:
  - Înregistrare mișcări pe x, y si z și pe baza interpretării la fiecare 30 de secunde am decis dacă individual se mișcă sau nu.
    - `float x = event.values[0];`
    - `float y = event.values[1];`
    - `float z = event.values[2];`

Implementare

Location Awareness

- Scop modul: folosit pentru a determina adresa IP a serverului pentru conectarea cu acesta.
- Componente :
  - Modulul senzorial : determină rețelele wireless
  - Modulul inteligent : compară rețelele cu cele din baza de date.
- Funcționare :
  - pentru determinarea rețelelor , pe PC: comanda **netsh** (Windows) , comanda **iwlist** (Linux)
  - pentru determinarea rețelelor, pe Android : serviciul **WIFI_SERVICE**
  - după determinarea rețelelor, datele sunt introduse în storage sub formă de ContextItem
  - modulul inteligent este notificat pentru a prelua datele din storage pentru comparare

Implementare

FileAnalyzer

- Scop : determină daca utilizatorul nu știe să-și implementeze problema, pentru a informa alți utilizatori din camera
- Componente :
  - **Modulul senzor** : determină fișierele schimbate din workspace pe baza datei ultimei modificări și dacă fișierele modificate mai sunt deschise
  - **Modulul inteligent** : analizează informațiile primite de la modulul senzor
- Funcționare :
  - data ultimei modificări este verificată regulat folosind un timer
  - verificare fișiere daca sunt încă deschise folosind **tasklist**
  - comparare număr de caractere scrise cu un prag
  - modificare interval interfață daca este nevoie de ajutor, după ultimele 2 răspunsuri

Implementare

PeerMachineManager

- Scop : trimiterea directă de mesaje între dispozitive, pe baza NetLink-ului
- Funcționare : preia din storage 3 tipuri :

- *other_devices_context* : lista de dispozitive a altui utilizator, primită de la server, pentru a le trimite *SEND_ITEM_CONTEXT*
- *send_item_context* : mesaj trimis tuturor dispozitivelor altui utilizator, pentru a cere ajutor
- *received_item_context* : mesaj trimis de alt utilizator, ca răspuns la send_item_context

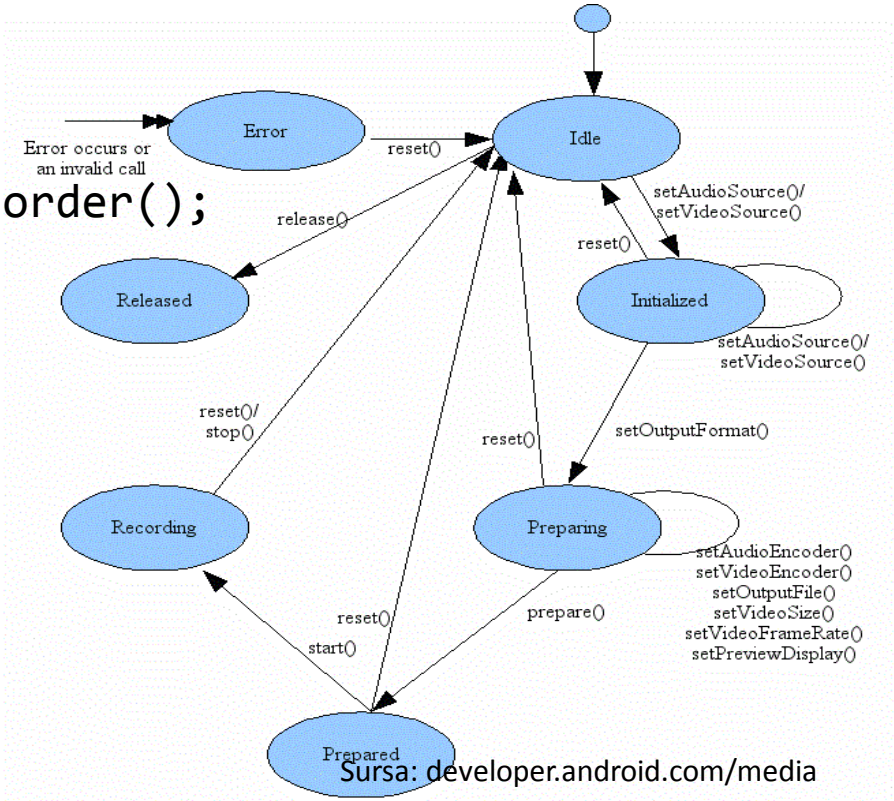
- În funcție de mesajul trimis, sunt afișate interfețe grafice, pentru a solicita ajutor, sau ca răspuns la solicitare

Implementare

Sunet

- Scop: înregistrare sunet la un interval si ajustarea volumului telefonului corespunzător
- Implementare: folosind clasa MediaRecorder

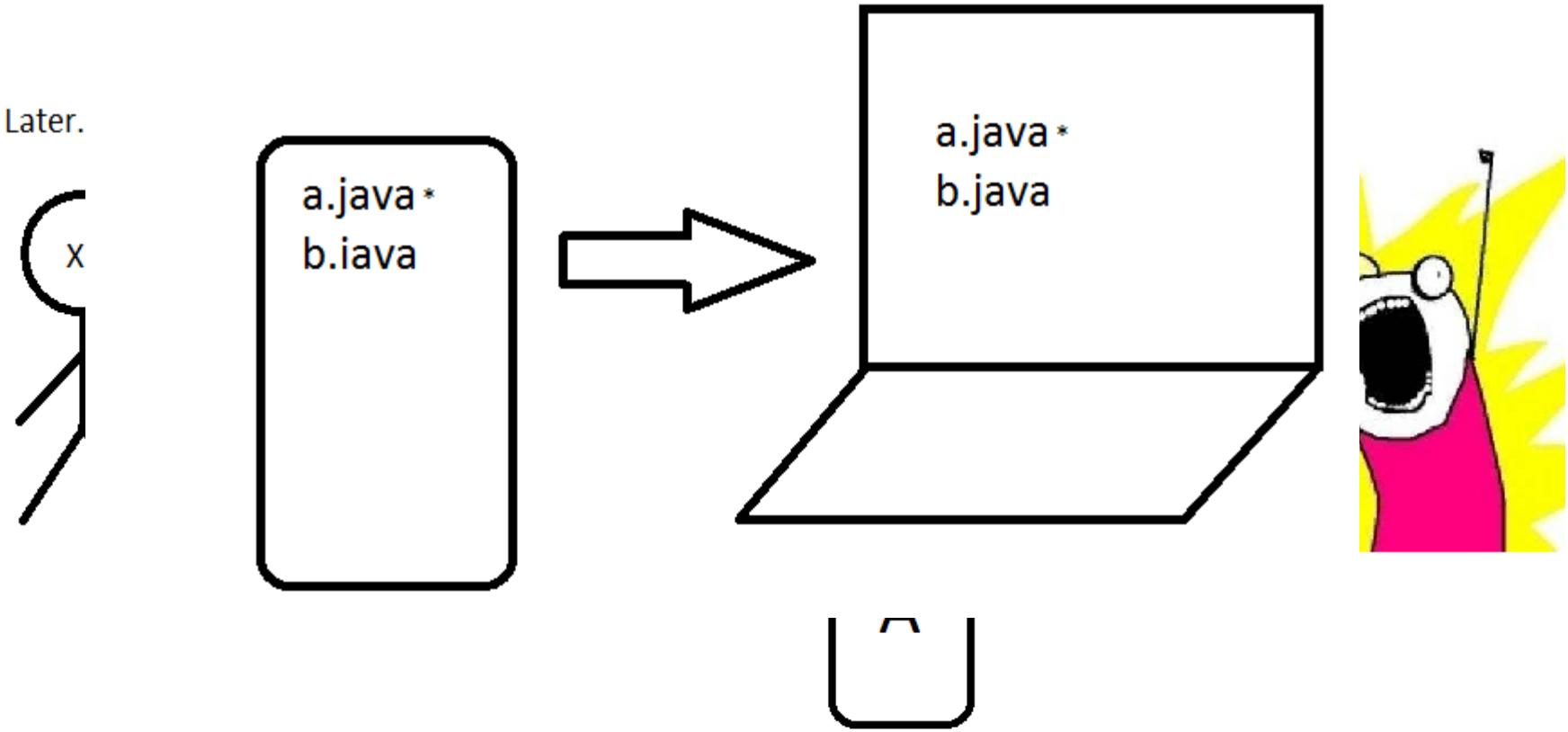
```
MediaRecorder m = new MediaRecorder();
m.setAudioSource(mic);
m.setOutputFormat(3gp);
m.setOutputFile("/ex.3gp");
m.setAudioEncoder(amr_nb);
m.prepare();
m.start();
m.stop();
m.release();
```



Sursa: developer.android.com/media

Scenariu

Animație



Concluzii

- Cunoștințe de Android, Eclipse & Java
- Familiarizarea cu Git
- Inteligența ambientală
- Lucrul cu un wiki
- Modularizarea implementării
- Javadoc, coding style
- Folosire comenzi shell
- Folosire noțiuni de rețele
- Research, StackOverflow
- Teamwork

Future work

- Adăugarea a noi module de inteligență, care pot lua decizii pe baza înregistrărilor făcute
- Crearea a noi senzori care să înregistreze alte activități
- Conectarea cu un dispozitiv Microsoft Kinect care poate determina poziția si postura
- Reprezentarea de către Android a profilului utilizatorului
- Înștiințarea fără cerere de către server despre noi utilizatori
- Schimb de informații despre evenimente din calendar
- Rulare aplicație in background, folosire notificări
- Cererea ajutorului persoanelor din alte rețele

Mulțumim

Întrebări?