Application Development for Mobile and Ubiquitous Computing

3. Adaptation and Context-Awareness

Dr. Ing. Thomas Springer
Technische Universität Dresden
Chair of Computer Networks
Recap

- **Mobile Computing**
  - any information, anywhere, at any time

- **Ubiquitous Computing**
  - the right information, at the right time, at the right place
Challenges:

- Connectivity
- Offline
- Form Factor
- Energy
- Usability
Lecture Structure

Application Development

Mobile Business Applications

Cross-Platform Development

Mobile Web Applications
Android
iOS
Windows Phone

Mobile Middleware

Disconnected Operations
Mobile Databases
Location-based Services
Communication Mechanisms

Enabling Technologies and Challenges
Motivating Example

- **Social Fitness App**
  - Wristband/Smart Watch + Smartphone/Tablet App
  - Server component for data storage and user management

- **Functionality**
  - Automatic detection and tracking of sports activities
  - Recording of activity states and content (images, videos, track record, curve with pulse, etc.)
  - Activity Timeline - Posting own activities and see others' activities in an integrated timeline with text, images, videos, etc.
  - Management of training schedule and planning of training activities
  - Managing competitions with ranking
Problem Analysis

- **Connectivity**
  - Low data rate and high delay in cellular networks
  - Intermittent connectivity due to weak connections and network changes

- **Offline Phases**
  - Limited coverage, unavailability of servers
  - To save energy, flat rate contingent, or communication cost

- **Form Factor**
  - Limited Resources (CPU, memory, display size)
  - Diversity of device classes (Smart Watches, Smartphones, Tablets) and device platforms (iOS, Android, Windows Mobile)

- **Energy**
  - Limited battery capacity
  - Display, computation, communication and sensing consume energy
Adaptation Examples

- Automatic detection and tracking of sports activities
  - Accelerometer based activity tracking
  - E.g. GPS is activated only if running or cycling activity to save energy
  - Higher sensing frequency is applied when walking or running is detected
- Sharing (upload and download) of activities to social network
  - Queueing for upload and download
  - Filtering (cycling related activities only), lossy conversion and lazy evaluation for data download
  - Prefetching and Caching of data (e.g. proactive download of activity content)
- Scheduling of training activities
  - Weather conditions monitored -> rescheduling outdoor activities if it is raining
- Network connection loss
  - Using of cached data for timeline, local storage of tracked data
What is a self-adaptive system?

"Self-adaptive software modifies its own behavior in response to changes in its operating environment. By operating environment, we mean anything observable by the software system, such as end-user input, external hardware devices and sensors, or program instrumentation." [Oreizy et.al]

Three mandatory requirements:

- **Context-Awareness:**
  - aware of operational environment
- **Self-Awareness:**
  - knowledge about current configuration or state
- **Self-Adaptation:**
  - modify itself at runtime
Application Development - 2. Adaptation and Context Awareness

- Self-Adaptivity
  - Context
    - What is context?
    - Location
    - Sensor Access
    - Context Abstraction
    - Characteristics of context
  - Adaptation Control
    - Intrinsic/extrinsic adaptation control
    - Feedback loops
  - Adaptation Mechanisms
    - Application data
    - Data transmission
    - Application structure
Knowing the Context makes the difference

- Hotel
  - Device Profile
  - User Profile
  - Context of use

- Office
  - Device Profile
  - User Profile
  - Context of use

- Customer visit
  - Device Profile
  - User Profile
  - Context of use

- Trip
  - Device Profile
  - User Profile
  - Context of use

- Application Server
  - WAN
  - WLAN + WAN
  - GSM/UMTS/Connexion

Dr. Thomas Springer
Application Development - 2. Adaptation and Context Awareness
Motivation

Context-aware Applications
Understanding of users current situation
Control of adaptation processes

- Social environment and activity
- Computing environment, device and network resources
- Mobility and location
- Interaction capabilities and user attention
- Situation/location dependent information
- Personalized and simplified user interactions
- Proactive, autonomous systems
- Adaptive system behaviour
Definition of Context

- “... all but the explicit input and output of an application” [LiS00]
- “... that which surrounds, and gives meaning to, something else” [o.V.00]

- “Context is a subjective concept that is defined by the entity that perceives it” and „contextual states [...] are inherently associated with specific objects” [Pascoe]

- „Context is an operational term: Something is context because of the way it is used in interpretation, not due to its inherent properties. [...] Features of the world become context through their use.” [Winograd]

- “Context is any information that can be used to characterise the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves” [Anind K. Dey]
## Types of Contextual Information

<table>
<thead>
<tr>
<th>Contextual dimension</th>
<th>Respective contextual information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical context</td>
<td>location, time, temperature, light and noise intensity</td>
</tr>
<tr>
<td>Technical context</td>
<td>network (bandwidth, latency, error rate), device (input and output capabilities, memory, software support), backend (available services, service preferences)</td>
</tr>
<tr>
<td>Personal context</td>
<td>Address, phone number, payment information, schedule,</td>
</tr>
<tr>
<td>Social context</td>
<td>Nearby persons, groups (teams) to which the user belongs, family and relatives, friends</td>
</tr>
<tr>
<td>Operational Context</td>
<td>Roles, activities, to-do-items, content of the inbox of the user</td>
</tr>
</tbody>
</table>
### Types Relevant for Example App

<table>
<thead>
<tr>
<th>Contextual dimension</th>
<th>Respective contextual information</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Physical context</strong></td>
<td>Location, time, temperature, light and noise intensity</td>
</tr>
<tr>
<td><strong>Technical context</strong></td>
<td>Network (bandwidth, latency, error rate), Device (input and output capabilities, memory, software support), available services, service preferences</td>
</tr>
<tr>
<td><strong>Personal context</strong></td>
<td>Address, phone number, payment, preferences, schedule, service preferences</td>
</tr>
<tr>
<td><strong>Social context</strong></td>
<td>Nearby persons, groups (teams) to which the user belongs</td>
</tr>
<tr>
<td><strong>Operational Context</strong></td>
<td>Roles, activities, to-do-items, content of the inbox of the user</td>
</tr>
</tbody>
</table>
Context in the Social Fitness App

Situation of interest:
User starts a workout, specifically running
Situation of interest:
User starts a workout, specifically running

Goto tracking mode:
- Activate GPS and set sensing frequency to 1s
- Start recording of locations + acc data
- Announce start of workout to training group
Context in the Social Fitness App

Situation of interest:
User starts a workout, specifically running

Action:
- Goto tracking mode:
  - Activate GPS and set sensing frequency to 1s
  - Start recording of locations + acc data
  - Announce start of workout to training group

Notify:

How can we find out, that the user starts running?
System Example: Technology for Enabled Awareness (TEA)

- Sensors on mobile device
- Goal: Sensing the situation of a mobile device and its user
  - Location and usage of mobile phone
  - Automatic selection of phone profile

- Used Sensors
  - two photodiodes
  - two microphones
  - dual axis accelerometer
  - digital temperature sensor
  - touch sensor
  - board extensible with further sensors (slots available on the board)
4-layered model:

1. Sensors (physical, logical)
2. Cue (processing of one sensor output)
   - contains array of values, operations on these values, e.g. average, standard deviation
3. Context (describes disjunct situations together with probability)
   - Description of the current situation on an abstract level
   - based on logic rules operating on cues
4. Application level (Situation and application specific)
   - definition of actions based on scripts, triggered if a certain situation is entered or left with a given probability
Cues

- Computing method: statistical functions

- Average value (for values of single sensor over about 1 min., applied for light, acceleration, temperature, and pressure)

- Standard derivation (for values over about 1 min., applied for light, passive IR, acceleration)

- Base frequency (applied for light and acceleration, types of light (flickering), acceleration pattern for moving)

- First derivative (applied for passive IR (Infra Red) and acceleration)
Exclusive contexts

• Rule set for recognizing the situations that device is situated in Hand, Table and Suitcase based on the sensors light, and acceleration in two directions (X and Y)
• Constants \( D_x, D_y, L, X_{\text{normal}}, Y_{\text{normal}}, D, \) and \( Q \) defined based on observations
• Location of mobile phone: hand/table/suitcase
• Device usage: inside/outside, car/bus/train, stationary/moving car

\[
\begin{align*}
\text{Hand}(t):&= \text{standard deviation}(\text{accel}X,t) > D_x, \\
&\text{standard deviation}(\text{accel}Y,t) > D_y, \\
&% \text{ device is slightly moving in X and Y} \\
&\text{average}(\text{light},t)>L. \% \text{ not totally dark} \\
\text{Table}(t):&= \text{abs}(\text{average}(\text{accel}X,t)-X_{\text{normal}})<D, \\
&\text{abs}(\text{average}(\text{accel}Y,t)-Y_{\text{normal}})<D, \\
&% \text{ the device is level in X and Y} \\
&\text{quartile}(\text{accel}X,t)<Q, \text{quartile}(\text{accel}Y,t)<Q \\
&% \text{ the device is stationary} \\
&\text{average}(\text{light},t)>L. \% \text{ not totally dark} \\
\text{Suitcase}(t):&= \text{average}(\text{light},t)<L. \% \text{ it is totally dark}
\end{align*}
\]
Application Scenario

- Environmental sound intensity controls volume of operating tones
- Font size, screen brightness and service content are adapted according to user activity and ambient light level

Used Sensors

- microphone, accelerometers, two channels for light, sensors for temperature, humidity, touch

Computed higher-level context

- **Location** \{**indoor**, **outdoor**\}
- **SoundType** \{**Car**, **Elevator**, **RockMusic**, **ClassicalMusic**, **TapWater**, **Speech**, **OtherSound**\}
1. Phase: abstraction of raw sensor data (comparable to cues)
- Crisp limits: true-false labeling of sensed data (e.g. Silent, Moderate, Loud - for environment sound intensity)
- Fuzzy sets: overlapping ranges (e.g. 0.7/Silent, + 0.3/Moderate+0/Loud)
2. Phase: computing higher-level context using naive Bayesian networks
Context Abstraction
Rule-based Approach

- Use Case: Smart Home
- Reasoning about users current situation at home
- Modeling:
  - CONON Ontology (CONtext ONtology)
  - OWL, RDF Triples
- Reasoning:
  - based on OWL light and RDF triples
  - Rule-based Reasoner (simple pattern matching algorithms)
temperature (12°) cold cloud formation, high probably of rain and storm
humidity (90%) high humidity
air pressure (1005 hPa) high air pressure

recommend user to reschedule cycling activity

Context Abstraction Process

C = ContextValue
A = Action

Context Capturing  Context Abstraction  Context Usage
Context Sources are:

- sensor devices (e.g. GPS, temperature, light intensity, noise)
- databases (e.g. extracting context from structured data in DB)
- application data (e.g. scheduling app)
- user monitoring and input (e.g. location or task from current user activity)
public class GpsActivity extends Activity {
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    LocationManager lMgr =
            (LocationManager)getApplicationContext().getSystemService(Context.LOCATION_SERVICE);
    LocationListener lListener = new LocationListener() {
        @Override
        public void onLocationChanged(Location l) {
            TextView tv = new TextView(GpsActivity.this);
            tv.setText("lat: " + l.getLatitude() + ",lon: " + l.getLongitude());
            setContentView(tv);
        }
    }
    lMgr.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, lListener);
}
}
Context Capturing Sensor Access in Android

- Access via the Android SDK
  - `android.hardware.SensorManager`
  - `sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);`
- Sensor supported?
  - `sensorManager.getSensorList(SensorManager.SENSOR_ALL)`

Observer Pattern:
- Register for the updates when Activity is in foreground
  ```java
  protected void onResume() {
    super.onResume();
    sensorManager.registerListener(accListener,
        sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER),
        SensorManager.SENSOR_DELAY_NORMAL);
  }
  ```
- `EventListener` observes changes in sensor data
  ```java
  class AccListener implements SensorEventListener {
    public void onAccuracyChanged(Sensor sensor, int accuracy) {...}
    public void onSensorChanged(SensorEvent event) {...}
  }
  ```
Abstracting low level context information

- accelerometer data record of a walk
- each axis is represented as a line
- significant pattern for different types of movement
  - e.g. pattern for steps on z-axis
- problems to consider:
  - result is axis dependent
  - gravity has impact on accelerometer
Step Detection via Accelerometer

- to reliably detect something, data processing can be crucial
- through
  - the root mean square over all 3 axt
  - removal of gravity
  - low pass filtering of the raw data
- a significant, orientation independent curve was created
Recognizing the activity

- Detecting steps based on peaks
- Derive:
  - No. of steps
  - Step frequency
  - With step length, distance can be derived
  - Speed

> Activity = Running if speed and step frequency are within certain thresholds
Properties of Context Attributes

- **Time dependent**
  - highest relevance at capturing time -> decreases constantly
  - History: represents values at different points in time

- **Location dependent**
  - highest relevance at capturing place -> decreases with distance
Properties of Context Attributes

**Imperfection**

- **Unknown** – sensor failure or unavailability
  - e.g., GPS does not work indoors

- **Erroneous** – due to measurement failures or wrong assumptions for derivation
  - e.g., no clear step pattern can be detected
  - e.g., wrong value for step length leads to wrong distance

- **Inaccurate** – due to uncertainty in measurements, application of heuristics
  - e.g., too many steps are counted due to wrong peek interpretation

- **Ambiguous** – due to conflicting alternative values
  - e.g., distance provided by acc and GPS is different
Goals:
- Provide Context information to applications
- Share context between applications

Two aspects:
- Architectures
- Context modelling
Why a Context Middleware?

- Reuse of generic functionality
  - Gathering, abstraction, storage, transport, application access
  - Mediates contextual information between context sources and context users
  - Enables sharing of context between applications
- Separation of concerns
  - Decoupling of applications and context sources
**Widgets:**
- Sensor abstraction
- Represent single context value
- Encapsulate details of sensors and other sources
- Current value, history, subscriptions

**Aggregator**
- Assign several context data (widgets, interpreters) to entities
- Easier management

**Interpreter**
- Processing of sensor data
- Generation of higher-level context
- One or several input values
- Can be used by Widgets, Aggregators, Interpreters and Applications
- System Building blocks are Sentient Objects (SO)
- Each SO represents context-aware functionality of gathering, processing or applying of contextual information
- Different types of objects
  - sensors
  - transformer
  - actuator
- Sentient objects contains internal representation of contexts as well as transformation code (e.g. inference mechanisms)
- Combination of sentient objects to context-aware applications
Context Middleware

Context Service: Architecture

- Application layer
  - Complex query service
  - Web service
  - Service
  - Web interface

- Context-framework layer
  - Query services
    - Location
    - Presence
    - Profile
    - Environment
  - Context interpreter
    - Reasoner
    - Knowledge base
    - Context model
    - Rules
  - Context providers
    - Location
    - Presence
    - Profile
    - Environment

- Context-gathering layer
  - Location system
  - Presence information system

- Device layer
  - RFID client
  - Wi-Fi client
5 layers:

- Modular design - Distributed deployment possible

1. Persistence layer
   - Persistent storage of context information (current and history)

2. Device layer
   - All devices delivering context information

3. Context-gathering components
   - Acquires special context information
     - Location, profile information, etc
   - Implements interfaces to context-framework layer

4. Context-framework layer
   - Acquires context information from previous layer and translates to OWL
   - Rule-based reasoning for deriving high level knowledge
   - Interface to Application layer

5. Application layer
   - Interfaces and web services for querying knowledge base
+ Server can provide high storage capacity and performance to process context
+ Easy to discover and manage
+ Central access control

- Single point of failure
- Limited scalability due centralized resource for high amount of clients
- Network connection and infrastructure required between mobile device and server for context access
Context Middleware
Decentralized Architecture

+ Independent from network connection to certain server
+ Support of ad-hoc scenarios
+ High scalability due to distribution of context access

- Availability of information depends on device availability, limited availability of user profiles
- Limited performance and storage capacity on mobile devices for context processing
- No central control to context access
Context Middleware
Two Categories of Information

- **State Information**
  - Application actively requests required context (pull)
  - Access to actual and historical data
  - E.g. current location, device, etc.

- **Change Events**
  - Application registers for particular change events, Waits passively for the events
  - Context service notifies registered applications about changes of state (push)
  - E.g. location changes, access network changes

<table>
<thead>
<tr>
<th>state</th>
<th>event</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>information</strong></td>
<td>current location of user, list of printers nearby</td>
</tr>
<tr>
<td><strong>action</strong></td>
<td>print a document using the closest printer</td>
</tr>
</tbody>
</table>
Context Middleware

Context Middleware API

Application

Consumer API

Internal Context Representation

Provider API

Context Source

Context Source

request

response

subscribe

publish

Shared understanding about context information required
Context Modelling

- Common understanding between context sources and context middleware
- Sharing of context information between context middleware and application
- Model for application development -> base for decision making

- Approaches
  - Context Profiles
  - Object Role Models
  - Ontology-based Models
**Name/Value Pairs:**
- username = „Thomas Springer“, temperature = 21 C

**Profiles**
- triples – subject, predicate, object
- `<http://www.examplepage.de>` - author – „Peter Müller“
- RDF (Resource Description Framework)
- CC/PP (Composite Capabilities/Preference Profiles)
- CSCP (Comprehensive Structured Context Profiles)
- Context is modelled based on facts, objects and roles
- Quality information can be added
- Model can be mapped to relational data model
Ontology-based Context Models

- "An ontology is an explicit specification of a conceptualisation" [Gru93]

- Describes real-world as formalized concepts
- Consists of concepts and roles (T-Box)
- Individuals as instances of concepts (A-Box)

Web Ontology Language (OWL)
- Defined by W3C
- Based on RDF
- 3 variants: OWL lite, OWL DL, OWL Full

\[
\text{Resident} = \text{Person and (livesAt some Home)} \\
\text{ResidentAtHome} = \text{Resident and (locatedAt some (Home or (isPartOf some Home)))}
\]
CONON – CONtext ONtology

Ref: Infocom Research, Singapore (Zhang)
Ref: Infocom Research, Singapore (Zhang)
Context Models - Comparision

- **Key/Value Pairs**
  - Most simple approach, easy to manage
  - Lack of structure and semantics

- **Context Profiles**
  - Hierarchically structured, Simple semantics
  - Mainly for static and rarely changing context -
    Does not support frequent updates
  - Goarse granularity for context exchange -
    whole profile or profile segments

- **Object Role Model**
  - Rich structure based on facts, objects and roles
  - Can be mapped to relational data models
  - Not widely adopted

- **Ontologies**
  - Can represent complex knowledge
  - Clear structure and reuse based on upper ontologies
  - Inference based on description logics reasoners
  - High complexity, performance might be weak
Context information enables awareness of execution environment

Pre-requisite to control adaptation processes
- Usually mapping between context information and control parameters/information necessary

The major phases of context processing
- Gathering
- Abstraction
- Decision making

Context models allow sharing of context

Context middleware provides general functionality to gather, process, store, distribute and access context information
Adaptation Mechanisms

Self-Adaptivity

Context

- Information gathered or extracted from different sources
- Provision via context middleware
- Defined with context model

Adaptation Control
- How to apply certain adaptation mechanisms in a detected context?

Adaptation Mechanisms
- Application data
- Data transmission
- Application structure

Dr. Thomas Springer
“Self-adaptive software modifies its own behavior in response to changes in its operating environment. By operating environment, we mean anything observable by the software system, such as end-user input, external hardware devices and sensors, or program instrumentation." [Oreizy et.al]

Three mandatory requirements:

- **Context-Awareness:**
  - aware of operational environment

- **Self-Awareness:**
  - knowledge about current configuration or state

- **Self-Adaptation:**
  - modify itself at runtime

![Diagram of Adaptive Software System with Self-Awareness, Self-Adaptation, and Context-Awareness](image)
Adaptation

- What is adapted to what?

- **Object** of Adaptation
  - application data
  - transfer of data
  - processing components (structure, placement)

- **Target** of adaptation
  - available resources of execution environment (device, network, ...)
  - user preferences, current situation

- Context information represents information about target of adaptation
  - necessary pre-requisites for adaptation
Adaptation of Application Data

- Preliminary consideration: data characteristics
  - Type (text, video, audio)
  - Format (.pdf, .mpeg, .wav, .mp3)
  - Coding (ASCII, Unicode UTF-8, ...)
  - Structure
    - atomic (e.g. image, logically atomic: input field and label)
    - tightly coupled (one file)
      - implicit structure (text document)
      - explicit structure (mime message)
    - loosely coupled (multiple files)
      - internal links (web content)
      - explicit structure description (I, P and B Frames of a video)
Adaptation of Application Data

- Important: adaptation of multimedia content
- Balance between performance, volume of data and lost information
  - Reduction – faster but coarse grained
  - Replacement and Transformation – slower but more fine grained
Goal: reduction of data volume
- Coarse grainded approach – keep or drop
- Low consumption of computation time and resources
  - no data processing but data selection and cancelation
- Filter rules for data selection defined
- Filtering possible based on different criteria
  - External criteria (dropping of packets in router according to threshold for queue length)
  - Data structure (dropping all email attachments, dropping all P and B frames of mpeg stream)
  - Data type (dropping videos in web pages)
  - Data format (dropping png images of web pages)
  - Semantics (dropping of private emails based on keywords)

Usually performed at runtime
Example: Activity Timeline

- **Header:**
  - Filter activities of particular type (e.g. cycling, outdoor)
  - Select activities for certain competition only (defined group of participants AND type of sports AND location)

- **Body:**
  - Filter activities by keywords in description (e.g. Team Challenge Dresden, new record)

- **Attachments:**
  - Discard attachments larger than a threshold (e.g. large images)
  - Discard all attachments of type video
- Goal: reduce data volume but limit content loss
- Operates on source data
- Creates related data objects of similar type
  - Keeping the content to some degree
- Operations dependent on data/media type
  - Change of specific properties of media objects
    - Image (resolution, colour depth)
    - Video (frame rate, size, colour depth)
    - Audio (sample rate, sample size)
    - Specific types (document, business object, etc.)

- More fine-grained reduction of media data dependent on device capabilities (e.g. display size or storage capacity) and connectivity
Pre-adaptation of data
- Creation of various quality levels for application data
- Client chooses quality level according to network capacity and device resources at runtime
- Dynamic changes between quality levels supported

+ No computing power consumed at runtime
  - High requirements for storage capacity limit number of versions
  - Limited flexibility for adaptation

Example: Video on demand (Youtube)
Reduction - Lossy Conversion
Adaptation of Application Data

- On-the-fly adaptation at runtime
  - Use current context (bandwidth, display size, etc.) to adjust parameters of data objects

  + Increased flexibility of adaptation
  + No additional storage requirements
    - Consumes processing resources at runtime
    - Can increase response time

- Example: Lossy conversion of image to reduce transfer size

  1600x1200 pixel  ➔  800x600 pixel  ➔  grayscale
Adaptation of Application Data

- Important: adaptation of multimedia content
- Balance between performance, volume of data and lost information
  - Reduction – faster but coarse grained
  - Replacement and Transformation – slower but more fine grained
Replacement

Goals:
- Reduce volume of data
- Keep information by replacement with synonym
- Replace source data with alternative data of **different type**
- Usually higher reduction of data volume (compared to lossy compression)

Selection
- Alternative data is explicitly available
  - Alternative text for image (ALT tag for IMG in HTML)
  - Replace image with file name

```html
<img src="img/pocketwatch0877.jpg"
alt="The image shows a young girl running across a bridge." />
```

The image shows a young girl running across a bridge.
- **Extraction**
  - Alternative data derived from source data
    - Extract representative image from video
    - Extract frame from image size

- **Generation**
  - Alternative data generated from source data
    - Speech-to-Text for audio data
    - Create vector image from pixel image

"Bla, bla"
Goal: change of data representation without information loss

Transformation

- Transformation of Coding
  - Recoding
  - Compression
- Format transformation
- Structure transformation
  - Structural changes
    - Composition
    - Decomposition
Goal: change internal representation of data without information loss
No change of data type, format and structure

Recoding
  • Change of internal data representation (avoid incompatibilities, increase efficiency)
    o Transformation of image data between color spaces
    o Recoding of ASCII-text to Unicode

Lossless Compression
  • Reduction of data size (entropy coding)
    o Generic mechanisms: Huffman, run length coding
    o Compression rate $\leq 1:8$
Goal: change of data representation without data loss to avoid incompatibilities

Change of format of source data

No change of data type

Examples:
- Change PDF documents to text representation
- Change PNG images to JPEG

Loss of information only if target format is less expressive
Transformation - Structure

- **Decomposition**
  - Goal: separate adaptation of fragments of structured data
  - Break down into fragments (atomic data objects)

- **Composition**
  - Goal: creation of compact representation from (atomic) data objects
  - reconstruct original data structure - inverse of decomposition
- **Structural Changes**
  - Goal: easier data processing, interoperability
  - Operation on structured data
  - Change XML document to DOM representation
  - Change implicit text document structure to explicit XML structure
Adaptation of Media Data

- Reduction of multimedia data with minimal information loss required
- Combination of different mechanisms (example JPEG)
  - Preparation - **decomposition** to 8x8 pixel blocks
  - Processing - **recoding** from space to frequency using Discrete Cosinus Transformation
  - Quantization – **lossy conversion** to reduce quality (accuracy) of information
  - Entropy encoding - **lossless compression**
Adapt the way data is exchanged between distributed components

- Transmission
  - Protocol parameters
  - Error handling
  - Prioritized data transfer

- Message delivery
  - Queuing
  - Bulking
  - Caching

- Rescheduling data access
  - Prefetching
  - Lazy evaluation
  - Delayed write-back
- **Protocol Parameters**
  - Frame size, packet size (maximum transfer unit)
  - Window size
  - Timeout
  - Header reduction
- **Error handling**
  - Error detection
  - Forward error correction
  - Optimized resending mechanisms
- **Prioritisation of data transfer**
  - Classification of data streams (interactive, background)
  - Assignment of priorities to classes or individual data streams
Adaptation of Communication

- Adapt the way data is exchanged between distributed components

  - Transmission
    - Protocol parameters
    - Error handling
    - Prioritized data transfer
  - Message delivery
    - Queuing
    - Bulking
    - Caching
  - Rescheduling data access
    - Prefetching
    - Lazy evaluation
    - Delayed write-back
Message Delivery - Caching

read access  read access  read access

App  Cache  Server

1: getFile(file1)  1: getFile(file1)

file1  file1

2: getFile(file1)

file1

3: getFile(file1)

file1

4: getFile(file1)

updated_file1
Message Delivery - Queuing

send message

forward message

Dr. Thomas Springer
Application Development - 2. Adaptation and Context Awareness
76
Message Delivery - Bulking

send messages

forward message set

App

Bulking

Unbulking

Server

1. sendMessage(msg1)
2. sendMessage(msg2)
3. sendMessage(msg3)
4. sendMessage(msg1+msg2+msg3)
5. sendMessage(msg1)
Adapt the way data is exchanged between distributed components.

- Transmission
  - Protocol parameters
  - Error handling
  - Prioritized data transfer
- Message delivery
  - Queuing
  - Bulking
  - Caching
- Rescheduling data access
  - Prefetching
  - Lazy evaluation
  - Delayed write-back
Rescheduling data access - Prefetching

prefetching

read access
Lazy Evaluation

ggetData  partialData  partialData  partialData

1: getData(entity1)
2: getFurtherData(entity1.2)
3: getFurtherData(entity1.3)

entity1.1 (partial result)
entity1.2
entity1.3

App  Proxy  Server
Lazy Evaluation - Example
Rescheduling Data Access

- Loading of the first three visible entries in the timeline only
- Only if user starts scrolling, the next entries in the timeline are loaded
- Details for entries, e.g. full video or full size images, are only loaded, if user selects detail view for timeline entry

Detail View

Load when scrolling

Dr. Thomas Springer
Application Development - 2. Adaptation and Context Awareness
Delayed Write-back

- readData
- write locally
- write back

Diagram showing the sequence of operations involving readData, write locally, and write back.
Rescheduling data access

- **Prefetching**
  - Transmission operation before explicit access
  - High availability, short response times
  - Efficient prediction mechanism required
    - Exploiting unused network capacity but waste of network capacity if prediction incorrect

- **Lazy evaluation**
  - Transmission operation after data access
  - Balancing network access, avoiding peaks
  - E.g.: loading of map tiles, display only visible part of web-pages

- **Delayed write-back**
  - Local changes to files only,
  - Write-back of files only if file is closed
  - Save bandwidth, avoid unnecessary updates
  - E.g.: distributed file system
Adaptation of Application Structure

- Filtering
- Conversion and compression of data
- Caching
- Prefetching

**Adaptation by Parametrization**
- Range of values (e.g., bandwidth)
- Mechanismus works over the whole range

**Adaption by Structural Changes**
- Range of values (e.g., error rate)
- Several mechanisms work within sub-ranges
- Changes of resource availability over sub-ranges require changes in application structure
Adaptation of application structure
Adaptation of application structure

- Place
- Connections
- Components

- Migration
- Creation
- Dynamic binding
- Indirection/proxy
- Branching
- Parallelization
- Fusion
- Adding
- Replacing
- Replication
Adapting Components

- **Adding/Delete**
  1. Add image scaling component to reduce data volume
- **Replacing**
  2. Replace component for image scaling with component for extracting image frames
- **Replication**
  3. Replicate image scaling component for load balancing
Adapting Connections

- Dynamic binding
  4. Rebind to an alternative proxy

- Branching
  5. Use alternative to balance load

- Parallelisation
  6. Use parallel pathes where copies of data are processed in different ways to provide multiple versions of content

- Indirection/Proxy
  - Use proxy pattern to decouple components
Adapting Placement

- **Creation**
  - Deploy components on appropriate host
  - Speech-recognition either at mobile device or on server (dependent on device resources)

- **Migration**
  - Change placement of components at runtime
  - Components "following" the user in the infrastructure
  - Handover application components from one mobile device to another one
Proxy Pattern

- basic architectural software pattern
- abstraction layer (intermediary) between communication partners
- indirect communication via intermediary
- goal:
  - introduction of additional functionality
  - proxy as a representative (reachability, act on behalf of component)
- can be used to extend existing applications and systems
Proxy on the Client

- Enables extensions of client functionality
- Support of disconnected operations
  - Proxy of server on the client (server emulation)
  - Caching, prefetching
- Adaptation of communication from client to server
- No data reduction for server-to-client communication
- Client not reachable during phases of disconnection
• represents client within the infrastructure
  • client reachable during disconnections
• independent from certain server
• data reduction for server-to-client communication
  • acts usually as generic proxy - generic adaptation support only
Proxy on Server

- enables extensions of server functionality
- data reduction for server-to-client communication
- application specific adaptation functionality
  - data filtering
  - adaptation of multimedia content
Proxy on Client and Server

- enables extensions on client and server for symmetric mechanisms
  - enhance communication protocols
  - error correction
  - encryption/decryption
  - encoding/decoding
Pipes and Filters Pattern

- Architectural pattern
- Goal: flexible configuration and reconfiguration of processing
- processing of a system decomposed into autonomous steps encapsulated in filters
- Loose coupling of filters - flexible selection of sequence of filters
- Increases reusability
- System consists of:
  - data source, chain of filters, data sink
  - filter contains typed interfaces for reading and writing data
  - pipes for interconnection of filters with interfaces of compatible types
  - push or pull exchange of data
Protocol Boosters

- Combination of adaptation mechanisms for communication protocols
- Booster
  - hardware or software module transparently injected into protocol processing
  - operate on particular base-protocol (e.g. TCP or UDP)
  - no change of base-protocol (transparent injection)
  - one booster: cumulation of TCP ACK, creation of duplicated ACK for TCP, implementation of Snoop protocol
  - pair of boosters: Forward error correction mechanism, encryption, compression
### Mechanisms for reducing and replacing data

<table>
<thead>
<tr>
<th>Mechanisms</th>
<th>Effect</th>
<th>Example</th>
<th>Placement</th>
<th>Number of Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filtering</td>
<td>Reduces amount of data and information, cancelation of not supported data types and formats</td>
<td>Attachments of an email, linked content of a web page (e.g. video data)</td>
<td>Proxy or Server, (before transmission)</td>
<td>One</td>
</tr>
<tr>
<td>Conversion</td>
<td>Reduces amount of data and information, adapts data content (data type specific)</td>
<td>Scaling of images, reduce framerate of video, sample rate of audio</td>
<td>Proxy or Server, (before transmission)</td>
<td>One</td>
</tr>
<tr>
<td>Selection</td>
<td>Selects one out of several alternative representations</td>
<td>Multiple versions of video, ALT tag in HTML for image text description</td>
<td>Proxy or Server</td>
<td>One</td>
</tr>
<tr>
<td>Extraction</td>
<td>Extracts alternative data, can change type of data</td>
<td>Frame from image size, file name of deleted attachment</td>
<td>Proxy or Server</td>
<td>One</td>
</tr>
<tr>
<td>Generation</td>
<td>Creates alternative data with same or reduced level of information, can change type of data</td>
<td>Speech to Text, generate vector representation from bitplane image</td>
<td>Proxy or Server</td>
<td>One</td>
</tr>
<tr>
<td>Mechanisms</td>
<td>Effect</td>
<td>Example</td>
<td>Placement</td>
<td>Number of Components</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------------------------------------</td>
<td>---------------------------------------------------</td>
<td>--------------------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>Transformation of coding</td>
<td>Changes coding of data, reduces amount of data (compression)</td>
<td>Change coding in GIF, ASCII to UNI code, Huffman coding for compression</td>
<td>Mobile Device and Proxy or Server</td>
<td>Two</td>
</tr>
<tr>
<td>Transformation of format</td>
<td>Changes format of data</td>
<td>GIF to JPEG</td>
<td>Mobile Device and Proxy or Server</td>
<td>Two</td>
</tr>
<tr>
<td>Transformation of structure</td>
<td>Changes structure of data</td>
<td>De/re-composition of email messages Web Pages (DOM – XML)</td>
<td>Mobile Device and Proxy or Server</td>
<td>Two</td>
</tr>
</tbody>
</table>
# Mechanisms for data transmission

<table>
<thead>
<tr>
<th>Mechanisms</th>
<th>Effect</th>
<th>Example</th>
<th>Placement</th>
<th>Number of Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adjusting protocol parameters</td>
<td>Improved bandwidth use, reduced response times, avoids overload</td>
<td>Timeouts, window size, packet size, reduced packet headers</td>
<td>Communication Partners</td>
<td>two</td>
</tr>
<tr>
<td>Forward error correction</td>
<td>No response channel required, no retransmissions</td>
<td>In combination to optimize balance between redundancy and resending (delay)</td>
<td>Communication Partners</td>
<td>two</td>
</tr>
<tr>
<td>Error correction ARQ</td>
<td>Performs retransmissions, no redundancy</td>
<td></td>
<td>Communication Partners</td>
<td>two</td>
</tr>
<tr>
<td>Prioritizing of data</td>
<td>Multiple service classes, optimized link usage</td>
<td>Prioritizing of web content, messages of different applications</td>
<td>Sender (and Receiver)</td>
<td>one (two)</td>
</tr>
</tbody>
</table>
## Adapting access time and message delivery

<table>
<thead>
<tr>
<th>Mechanisms</th>
<th>Effect</th>
<th>Example</th>
<th>Placement</th>
<th>Number of Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caching</td>
<td>Reduces response time and amount of data, increased availability</td>
<td>Coda (disconnected op.), Web Caches</td>
<td>Mobile Device or Proxy</td>
<td>One</td>
</tr>
<tr>
<td>Bulking</td>
<td>Increase throughput (and delay)</td>
<td>Bulking short RPC request messages</td>
<td>Mobile Device and Server</td>
<td>Two</td>
</tr>
<tr>
<td>Queuing</td>
<td>Increase autonomy, increases response time, temporal storage</td>
<td>Queued-RPC, Message oriented Middleware</td>
<td>Mobile Device and Server</td>
<td>One</td>
</tr>
<tr>
<td>Prefetching</td>
<td>Reduces response time, increased availability, waste of bandwidth</td>
<td>Web content, files, data</td>
<td>Mobile Device or Server</td>
<td>One</td>
</tr>
<tr>
<td>Lazy evaluation</td>
<td>Reduces amount of data to be transferred</td>
<td>Outlines for Web content</td>
<td>Mobile Device and Proxy or Server</td>
<td>One</td>
</tr>
<tr>
<td>Delayed write-back</td>
<td>Decreases amount of data/messages, distributes network load over time, increases conflict probability</td>
<td>Coda (disconnected operation), Read/Write Caches</td>
<td>Mobile Device and Proxy or Server</td>
<td>One</td>
</tr>
</tbody>
</table>
Adaptation Control

Self-Adaptivity

Context

Context Awareness
- Information gathered or extracted from different sources
- Provision via context middleware
- Defined with context model

Adaptation Control
- How to combine adaptation mechanisms in an App?
- How to apply certain adaptation mechanisms in a detected context?

Control

Mechanisms

Adaptation Mechanisms
- Application data
- Data transmission
- Application structure
MAPE-K feedback loop as architectural reference for designing self-adaptive software systems

- **M** – Monitor
  - Collecting data from sensors and further data sources
  - Correlating measurements to higher level context

- **A** – Analyze

- **P** – Plan

- **E** – Execute

- **K** – Knowledge

**Example:**
- Monitoring of network connection, user activity, weather conditions, location
- **MAPE-K feedback loop** as architectural reference for designing self-adaptive software systems

- **M** – Monitor
- **A** – Analyze
  - Evaluation of sensed values to determine if system needs to be changed
- **P** – Plan
- **E** – Execute
- **K** – Knowledge

- Example:
  - Network connection lost AND reestablishment fails
  - Situation: Offline work
MAPE-K feedback loop as architectural reference for designing self-adaptive software systems

- M – Monitor
- A – Analyze
- P – Plan
  - Reflecting on current system state and identifying appropriate state changes
  - Definition of adaptation plans
- E – Execute
- K – Knowledge

Example:
- Situation: Offline work
  - Establish local tracking, use cached data for timeline, activate queueing
MAPE-K feedback loop as architectural reference for designing self-adaptive software systems

- M – Monitor
- A – Analyze
- P – Plan
- E – Execute
  - Execution of adaptation decisions on target system
  - Change of system configuration
- K – Knowledge

Example:
- Establish local tracking, use cached data for timeline, activate queueing
  - Execute parameter and structural changes
MAPE-K feedback loop as architectural reference for designing self-adaptive software systems

- M – Monitor
- A – Analyze
- P – Plan
- E – Execute
- K – Knowledge
  - Stores information about the managed element, e.g., runtime or context model(s), history of collected sensor information etc.

Example:
  - Update knowledgebase to reflect executed changes
Internal and External Adaptation Control

2. Adaptation and Context Awareness

Self-Adaptive Software

Sensing

Effecting

Managed Application

Adaptation Management

Effecting

Dr. Thomas Springer
- use of an external adaptation engine
- clear separation of concerns
Level of Adaptation

- **Laissez-faire**
  - No system support for adaptation
  - Applications have to explicitly implement adaptation functionality
  - Application controls resource allocation and adaptation
  - No optimization of resource allocation and adaptation for concurrent applications

- **Application-transparent**
  - No application changes necessary
  - Centralized adaptation control and resource allocation by system for concurrent applications
  - No application level information usable for adaptation
  - No application specific adaptation functionality
Application-aware adaptation

- Co-operation between system and application
- Combination of advantages
  - application specific adaptation operations
  - application-level information for adaptation control
  - central coordination of resource allocation and adaptation

Client

application

Interceptor

resourcemanager (Viceroy)

warden1

warden2

warden3

server

several levels of quality for data objects

Dr. Thomas Springer

Application Development - 2. Adaptation and Context Awareness

Noble, Satyanarayanan (Carnegie Mellon University)
Adaptation as a principle how to build applications
- Mechanisms are adoptable in different contexts
- Large variety of adaptation mechanisms available
  - data adaptation
  - communication
  - application structure
- Mechanisms can be combined
- Adaptation Control based on feedback loop (MAPE-K)
References

- T. Springer and A.-Y. Turhan: Employing Description Logics in Ambient Intelligence for Modeling and Reasoning about Complex Situations. Journal of Ambient Intelligence and Smart Environments (JAISE), 235 - 259 , 2009