Vikas Agarwal, **Sunil Goyal**, Sumit Mittal, Sougata Mukherjea
IBM Research – India
2nd December 2009

MobiVine –

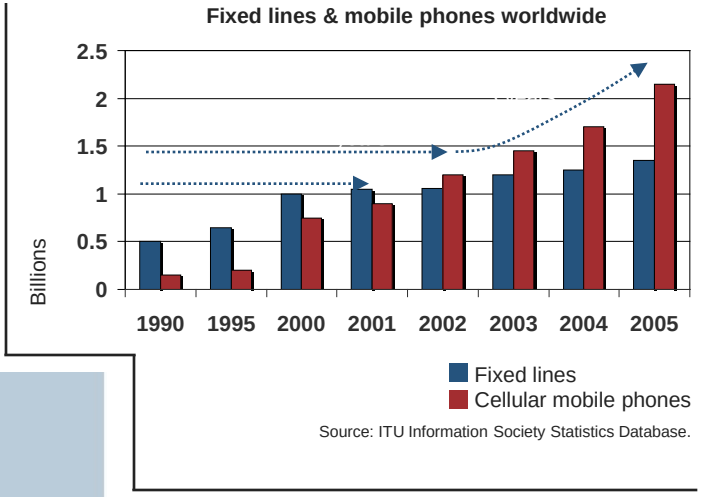
A Middleware Layer to Handle Fragmentation of Platform Interfaces for Mobile Applications

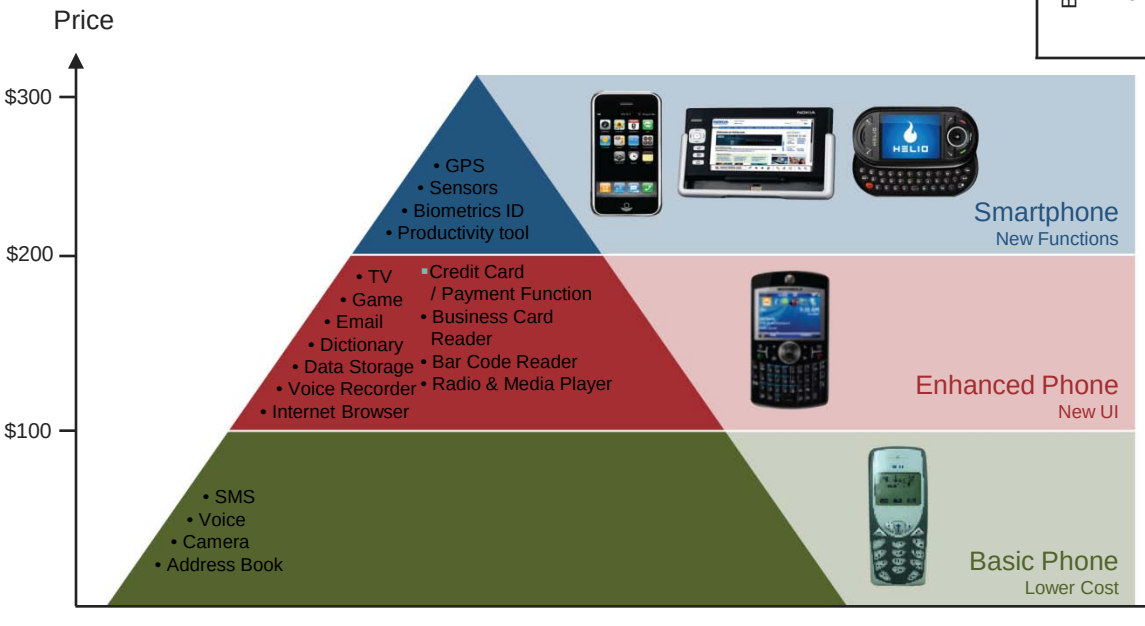
Mobile Phone as A Platform

Growing computing power, functions and dropping cost make mobile phone capable as an alternative to PC to access information and services

Mobile phone information service platform

- Growing 1 Billion per year through 2011
- 3x the number of PCs today
- 2x the number of credit cards today
- 2x the number of TVs today





- **Mobile phone: an alternative to PC**
- Young generation
- Leading markets (e.g. Japan, Korea)
- The base of pyramid (e.g. India, Africa)

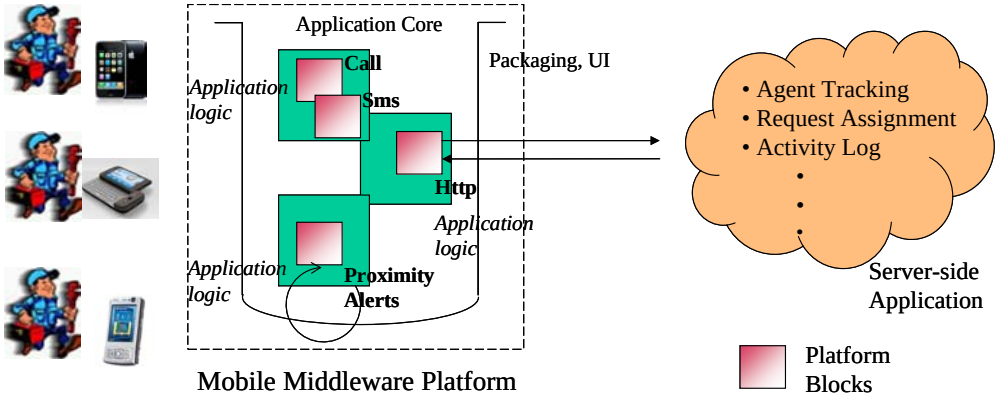
Motivation

- As the new killer application remains elusive…differentiation will come from ability to rapidly compose customized services for narrow customer segments
  - This must be independent of platforms (Nokia S60, Apple iPhone, BlackBerry, Android, …)
- Enable smarter and richer applications
  - Rich Client Device Features (PIM, Bluetooth, Camera, …)
  - Telecom Infrastructure Services (Location, Presence, 3PCC, …)
  - 3rd party services (Maps, News feeds, …)
- Accelerate application development on emerging smart phone platforms
  - Application development must
    - Provide Mobile Device Breadth
    - Reduce Cost of development
    - Provide Ease of deployment & update
    - Allow easy integration of Device and Network Functions

Challenges

- Consider development of a mobile application
  - Key components
    - UI
    - Application Logic blocks
    - Platform blocks

      Integrated using platform interfaces
  - Packaged as per platform requirement



  - Application provider desire to roll out app on multiple platforms
    - Significant porting effort, specially for Platform blocks, because

      Interfaces for similar function across platforms are different

      Syntax variation (Parameter name, data type, ordering, Exceptions)

      Semantic differences

```
LocationManager.addProximityAlert (double latitude, double longitude,
float radius, long expiration, Intent intent) throws SecurityException
```

v/s

```
LocationProvider.addProximityListener (ProximityListener listener,
Coordinates coordinates,
float proximityRadius) throws SecurityException, LocationException,
IllegalArgumentException, NullPointerException
```

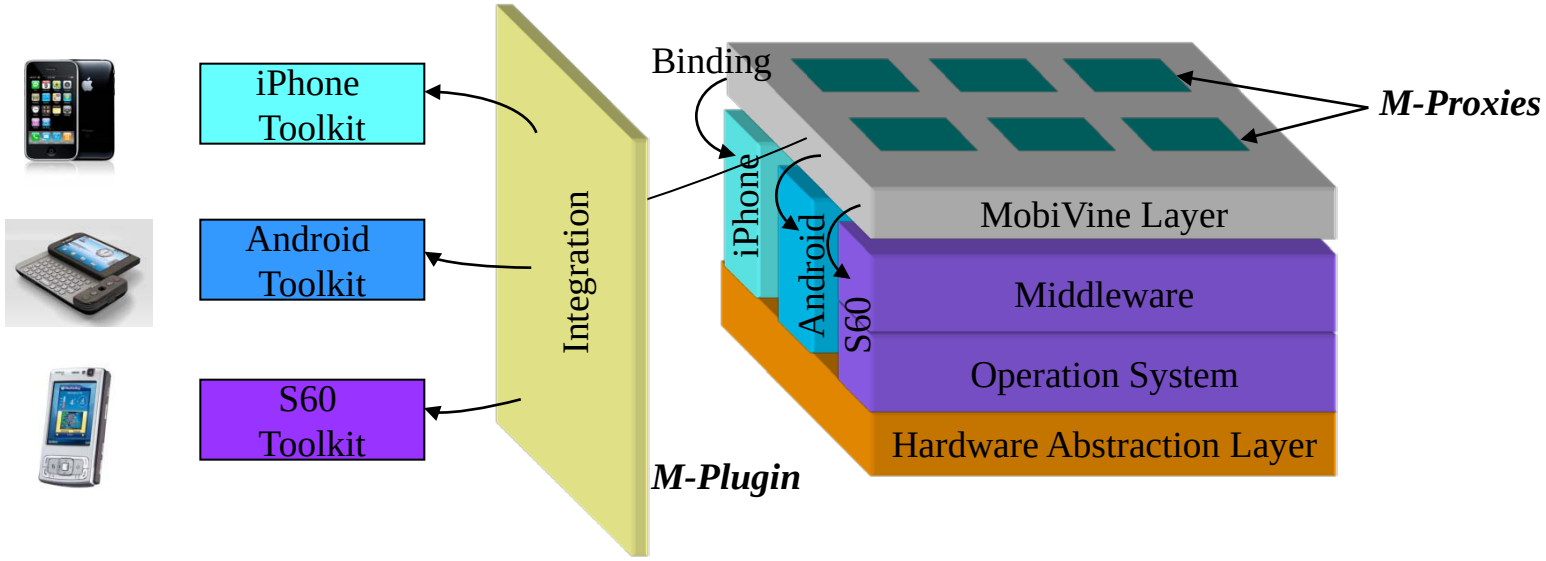
Contribution

- Study of the problem of application fragmentation due to heterogeneity in platform interfaces
- We propose ‘MobiVine’, a middleware layer to handle fragmentation issues for mobile applications
  - M-Proxy, for abstracting the platform interface heterogeneity
  - M-Plugin, for seamless integration of the model with existing toolkits
  - Caters to various different categories of developers
    - Java, JavaScript, BPEL, etc.
- Implementation
  - Prototype based on suggested architecture
  - Demonstration of a Mobile Application Development using MobiVine toolkit

Rest of the Talk

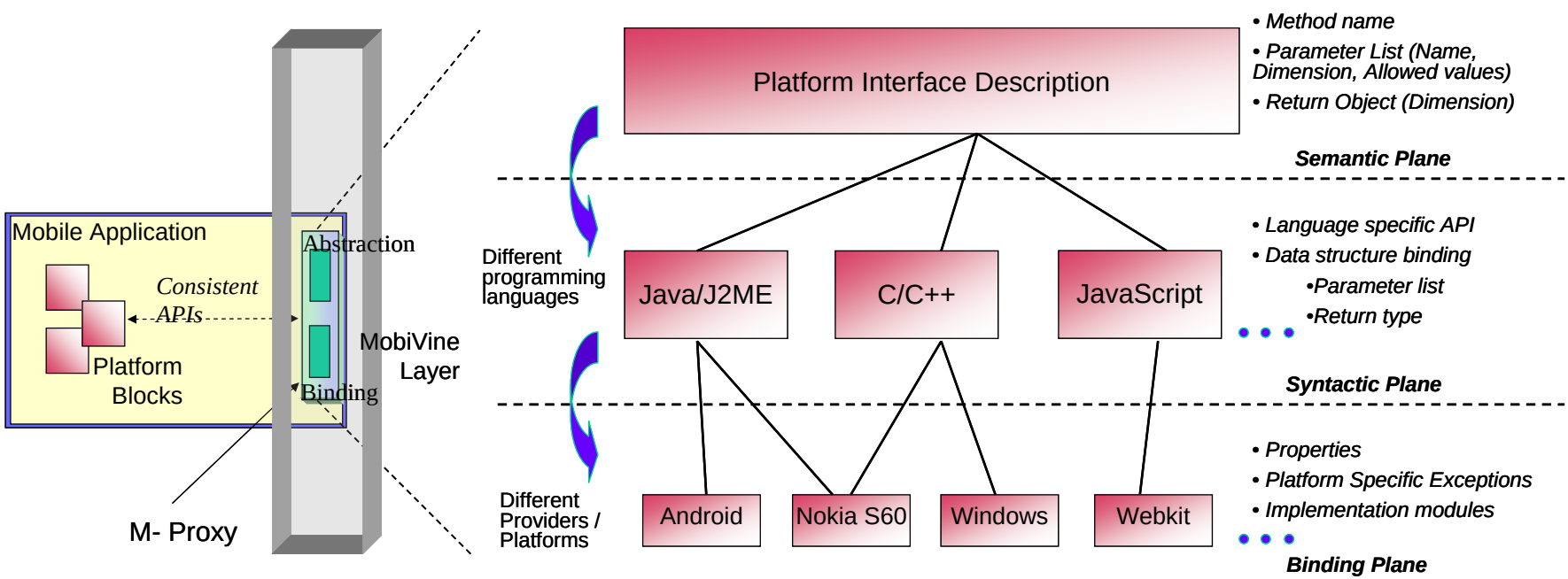
- MobiVine Middleware
- M-Proxy Model
- M-Plugin
- Implementation
  - MobiVine Implementation
  - Prototype based on suggested architecture
- Evaluation
- Discussion and Conclusion

MobiVine Overview



- "MobiVine", a middleware layer to handle fragmentation of platform interfaces for mobile applications, consists of two main components.
  - *M-Proxies*, a semantically structured unit which helps abstract the platform interface heterogeneity and fragmentation across different platforms while binding to the underlying middleware stacks, and
  - *M-Plugins*, which integrates MobiVine with existing tooling & deployment infrastructure while utilizing the information kept in a structured format inside M-Proxies.

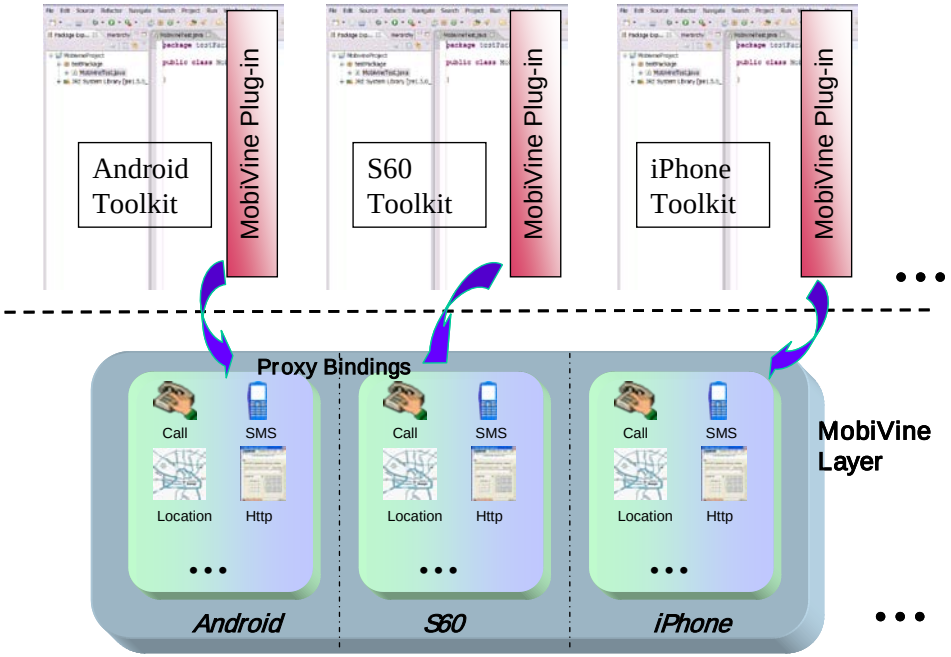
M-Proxy Model



M-Proxy Features

- Reduce Platform fragmentation
  - Common APIs for different platforms
    - Semantics, syntax, consistent data structures
  - Cleaner code, easy to port from one platform to another
- Richness
  - Include platform specific attributes and properties
    - Default and allowed values
  - Desired by the developer – number of retries for sending SMS, making calls
  - Output format of location information
- Is extensible
  - for new functions, additional programming languages, new platforms
- Easily pluggable into Existing Toolkits
  - Structure of the proxies helps

M-Plugin



- Developers tend to (and like to) use standard application IDEs
- M-Proxies can be integrated into existing toolkits using M-Plugins
- M-Plugin supports
  - M-Proxy visibility
  - M-Proxy Presentation
  - M-Proxy Configuration
  - M-Proxy Embedding
- An M-plugin is required for each supported platform

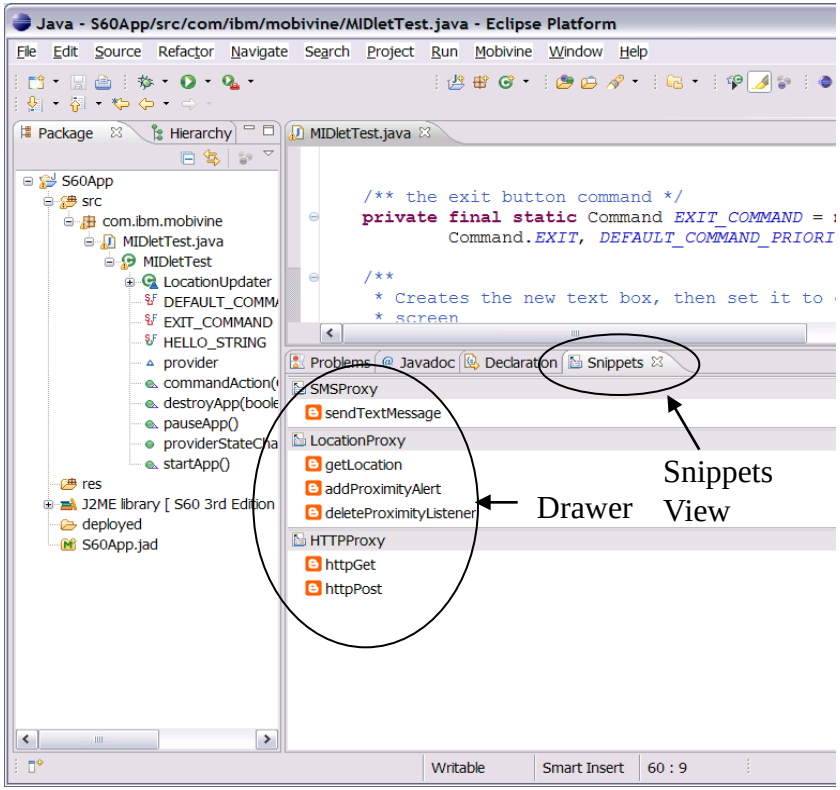
Implementation (1 of 2)

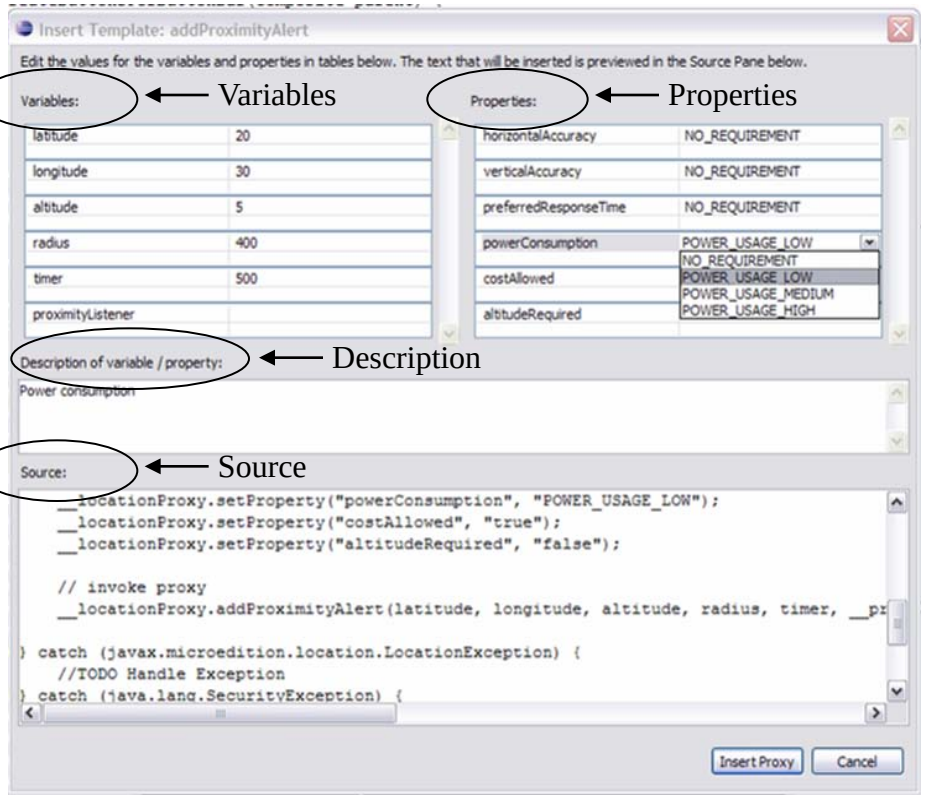
- Implemented for three platforms
  - Nokia S60
  - Andrioid
  - Android WebView
- And for two programming languages
  - Java (Nokia J2ME MIDlets, and Android Project)
  - JavaScript (WebKit based web application)
- M-Proxies
  - Five XML schemas captures each function on these platforms across two languages
- Function proxies developed for Location, SMS, Call, and Http
  - Using Android SDK release m5-rc15
  - And Nokia S60 Third Edition SDK
- M-Plugins
  - As an extension to Eclipse (for both Android and S60)

Implementation (2 of 2)

- Several Complexities
  - Handling platform specific attributes
  - Handling callbacks on Android
  - Enabling notion of JavaScript proxy object that interacts with underlying Java proxy object
  - Providing support for callbacks in JavaScript objects
    - Now standard part of Android OS since version 1.0
- Demonstration
  - “Field Agent Management” application

MobiVine Toolkit for for S60 Platform



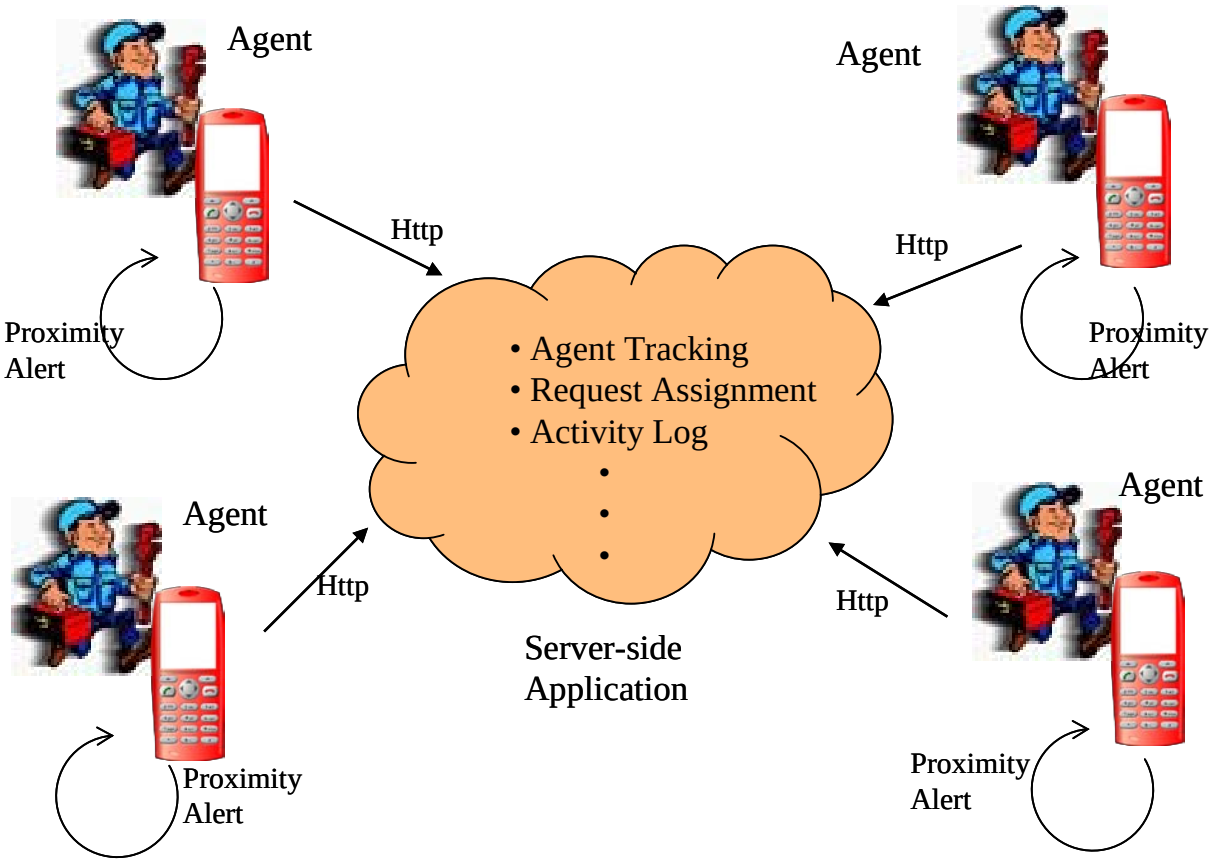


Application

- Implemented in Java for S60
- Ported on Android in Java
- And Implemented in JavaScript for Android WebView

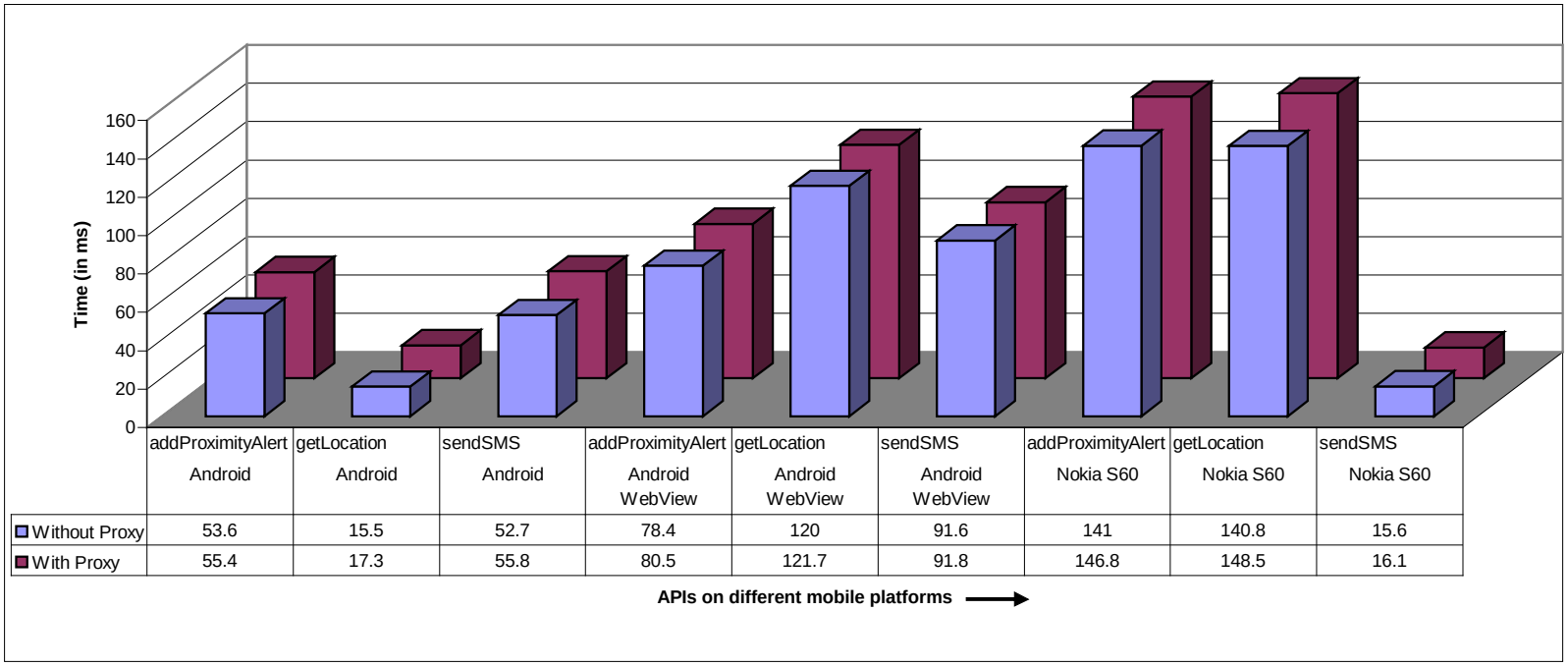
- Integration of device services – GPS location, Proximity Alert, SMS and Http – could be completed in an hour using MobiVine toolkit

- Porting was less than 15 minutes for device functions



Discussion

- The notion of M-Proxy makes mobile application code
  - Easily portable across multiple mobile platforms
  - Less complex to develop and debug
  - Easier to maintain and migrate to new version of the platform
  - Minimally affected from performance perspective, as proxy overhead is minimal



	addProximityAlert Android	getLocation Android	sendSMS Android	addProximityAlert Android WebView	getLocation Android WebView	sendSMS Android WebView	addProximityAlert Nokia S60	getLocation Nokia S60	sendSMS Nokia S60
Without Proxy	53.6	15.5	52.7	78.4	120	91.6	141	140.8	15.6
With Proxy	55.4	17.3	55.8	80.5	121.7	91.8	146.8	148.5	16.1

Summary

- We Presented MobiVine, a middleware layer to handle fragmentation of mobile device platform interfaces for development of Mobile applications.
- The core architecture component called ‘M-Proxy’ helps abstract heterogeneities in interfaces across different platforms while binding to the underlying middleware stack.
- The other component called ‘MPlugins’ helps integrate MobiVine with existing development tools and deployment infrastructure
- With the help of a prototype implementation for three platforms - Android, S60 and Android WebView, we evaluated the effectiveness of our approach based on various software engineering principles and performance metrics.

- Future Work
  - Add proxies for more functions
    - specially related to data like Calendar, and Address Book
  - UI Concern
  - Offering mashup tools
    - that allow integration of Telecom Network functions and Device functions leveraging M-Proxy model and M-Plugin approach

Contact:

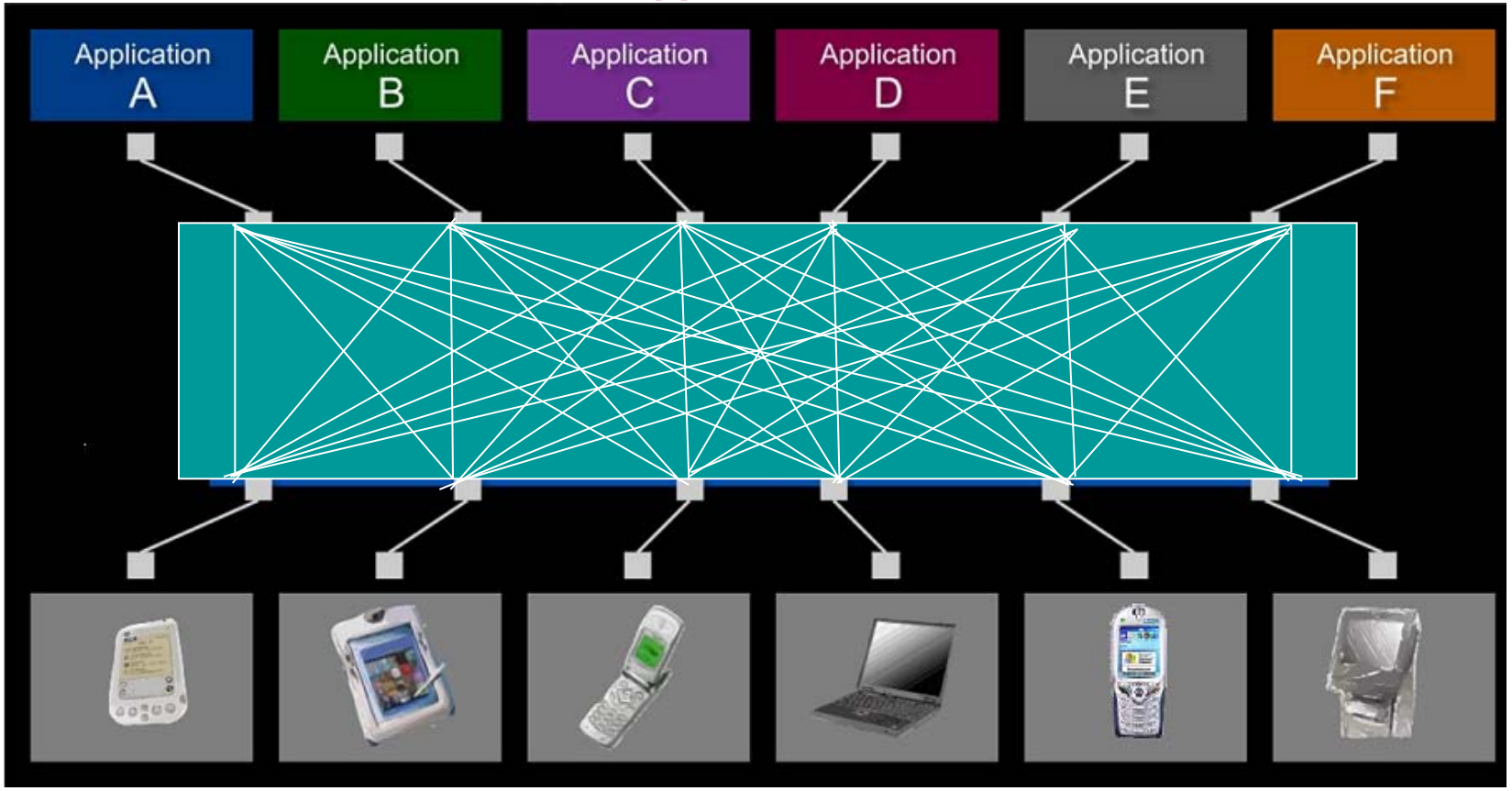
Sunil Goyal (gsunil@in.ibm.com)

IBM Research – India, New Delhi

+91-11-41292100, 6619 2100

The Problem: Beyond the classic PC, reality is complex…

M applications ...



N devices

How do you solve an expanding "M x N" matrix?

M-Proxy Schema Representation

Code Fragment (without using proxy)

```
public class WorkForceManagement extends Activity {
    class ProximityIntentReceiver extends IntentReceiver {
        double latitude;
        double longitude;

        public ProximityIntentReceiver(double latitude, double longitude) {
            this.latitude = latitude;
            this.longitude = longitude;
        }

        public void onReceiveIntent(Context ctxt, Intent i) {
            String action = i.getAction();
            if (action.equals(PROXIMITY_ALERT)) {
                boolean entering = i.getBooleanExtra("entering", false);
                LocationManager lm = (LocationManager)
                ctxt.getSystemService(Context.LOCATION_SERVICE);
                Location loc = lm.getCurrentLocation("gps");

                /* business logic for handling proximity events */
            }
        }
    }

    static final String PROXIMITY_ALERT =
"com.ibm.proxies.android.intent.action.PROXIMITY_ALERT";
    ...
    ...
    public void onCreate( .... ) {
        // registering for proximity events

        Context context = this;
        try {
            ProximityIntentReceiver proximityReceiver =
                new ProximityIntentReceiver(latitude, longitude);
            context.registerReceiver(proximityReceiver,
                new IntentFilter(PROXIMITY_ALERT));
            LocationManager lm = (LocationManager)
                context.getSystemService(Context.LOCATION_SERVICE);
            Intent i = new Intent(PROXIMITY_ALERT);
            lm.addProximityAlert(latitude, longitude, radius, timer, i);
        } catch (Exception e)
            // Handle Android specific exception
        }
        ...
    }
}
```

```
public class WorkForceManagement extends MIDlet implements
            ProximityListener, LocationListener {
    float radius;
    Coordinates coordinates = null;
    boolean entering = false;
    long startTime, timeOut;
    LocationProvider lp;

    public void proximityEvent(Coordinates coordinates, Location lo) {
        long currentTime = System.currentTimeMillis() / 1000;
        if ((currentTime - startTime) > timeOut) {              //time out
            lp.setLocationListener(null, -1, -1, -1);
            LocationProvider.removeProximityListener(this);
            return;
        }
        entering = true;
        //business logic for entry event
    }

    public void locationUpdated(LocationProvider lp, Location lo) {
        long currentTime = System.currentTimeMillis() / 1000;
        if ((currentTime - startTime) > timeOut) { //time out
            lp.setLocationListener(null, -1, -1, -1);
            LocationProvider.removeProximityListener(this);
            return;
        }

        if (entering == false)
            return;
        float distance = getDistance(coordinates, lo);
        if (distance > radius) {
            entering = false;
            //add business logic for exit event
            try {
                // registering for proximity events
                LocationProvider.addProximityListener(this, coordinates, radius);
            } catch (Exception e) {
                // Handle S60 specific exceptions
            }
        }
    }

    public void startApp() {
        // registering for proximity events
        this.radius = radius;
        this.coordinates = new Coordinates (latitude, longitude, (float)altitude);
        this.timeOut = timeOut;
        this.startTime = System.currentTimeMillis() / 1000;
        try {
            criteria.setPreferredResponseTime(Criteria.NO_REQUIREMENT);
            criteria.setVerticalAccuracy(50);
            lp = LocationProvider.getInstance(criteria);
            lp.setLocationListener(this, -1, -1, -1);
            LocationProvider.addProximityListener(this, coordinates, radius);
        } catch (Exception e) {
            // Handle S60 specific exceptions
        }
        ...
    }
}
```

Code Fragment (using proxy)

```
public class WorkForceManagement extends Activity
                implements ProximityListener {
    ...
    public void onCreate( ... ) {
        // registering for proximity events
        try {
            LocationProxyImpl loc = new LocationProxyImpl();
            loc.setProperty("context", this);
            loc.setProperty("provider", "gps");
            loc.addProximityAlert(latitude, longitude, altitude, radius,
                timer, this);
        } catch (Exception e) {
            // Handle Android specific exceptions
        }
        ...
    }

    public void proximityEvent(double refLatitude, double refLongitude,
        double refAltitude, Location currentLocation, boolean entering) {
        /* business logic for handling proximity events */
        ...
    }
}
```

```
public class WorkForceManagement extends MIDlet
        implements ProximityListener{
     ...
    public void startApp(...){
        //registering for proximity events
        try {
          LocationProxyImpl loc = new LocationProxyImpl();
          loc.setProperty(..);
          loc.addProximityAlert(latitude, longitude, altitude, radius,
                timer, this);
        } catch (Exception e) {
            // Handle S60 specific exception
        }
          ...
     }

    public void proximityEvent(double refLatitude, double refLongitude,
        double refAltitude, Location currentLocation, boolean entering) {
        /*business logic for handling proximity events*/
        ...
    }
    ...
}
```

Code Fragment (using proxy)

```
<script type="text/javascript" >
    function JSInit(...) {
        try {
            // registering for proximity events
            var loc = new LocationProxyImpl();
            loc.setProperty("provider", "gps");
            loc.addProximityAlert(latitude, longitude, altitude, radius, timer, proximityEvent);
        } catch (ex) {
            // Handle Android specific exceptions
        }
        ...
    }

    function proximityEvent(refLatitude, refLongitude, refAltitude, currentLocation, entering) {
        /* business logic for handling proximity events */
        ...
    }
    ...
</script>
```