# Smart Composition of Reusable Software Components in Mobile Application Product Lines

**Authors:**

Ricardo Erikson V. de S. Rosa, UFAM

**Vicente F. Lucena Jr, UFAM**

# Contact Profile

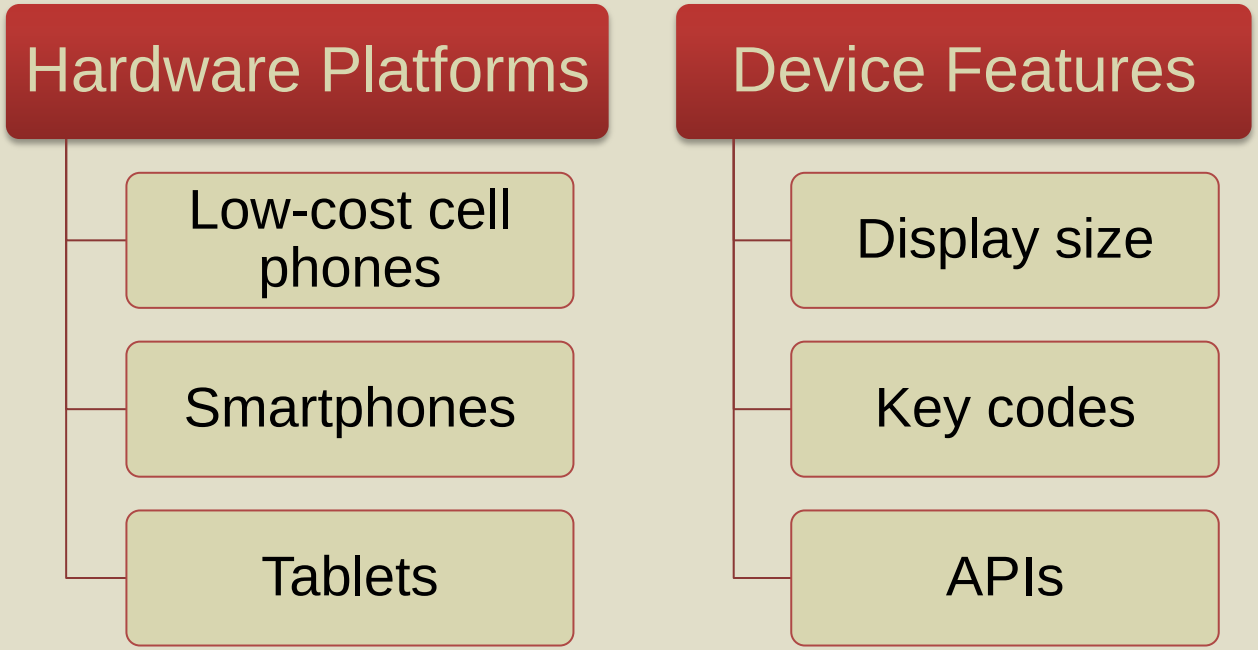

- Ricardo Erikson V. de S. Rosa
  - *ricardoerikson@ufam.edu.br*
- Vicente F. Lucena Jr, UFAM
  - *vicente@ufam.edu.br*

# Agenda

- The Problem
- Current Solutions
- Our Solution
- Why is it Interesting?
- Conclusions

# The Problem

- The development of mobile applications presents several challenges.



# The Problem

Touch screen

Accelerometer sensor

QWERTY keyboard

360x640 screen resolution

Bluetooth



Bluetooth

Numeric keyboard

320x240 screen resolution

# The Problem

- The high heterogeneity in mobile devices features may lead to a significant increase in software variability.
- The developers need to create several versions of their applications.
- The development of mobile applications becomes more complex.

# Current Solutions

- Commercial solutions:
  - pure::variants.
  - Gears.
- Constraint Satisfaction Problem (CSP).
- The current solutions do not fully satisfy the requirements of the mobile application domain.

# Our Solution

- The selection mechanism is based on a search engine of software components.
- An XML based approach was used to represent the SPL composition.
- Dependency Injection on the mobile device.

# Our Solution

```
<componentData>
    ...
    <component id="01" class="MainApp">
        <composite property="sensor" cmpref="02" />
        <composite property="menu" cmpref="03" />
        <composite property="menu" cmpref="04" />
    </component>
    ...
</componentData>
```

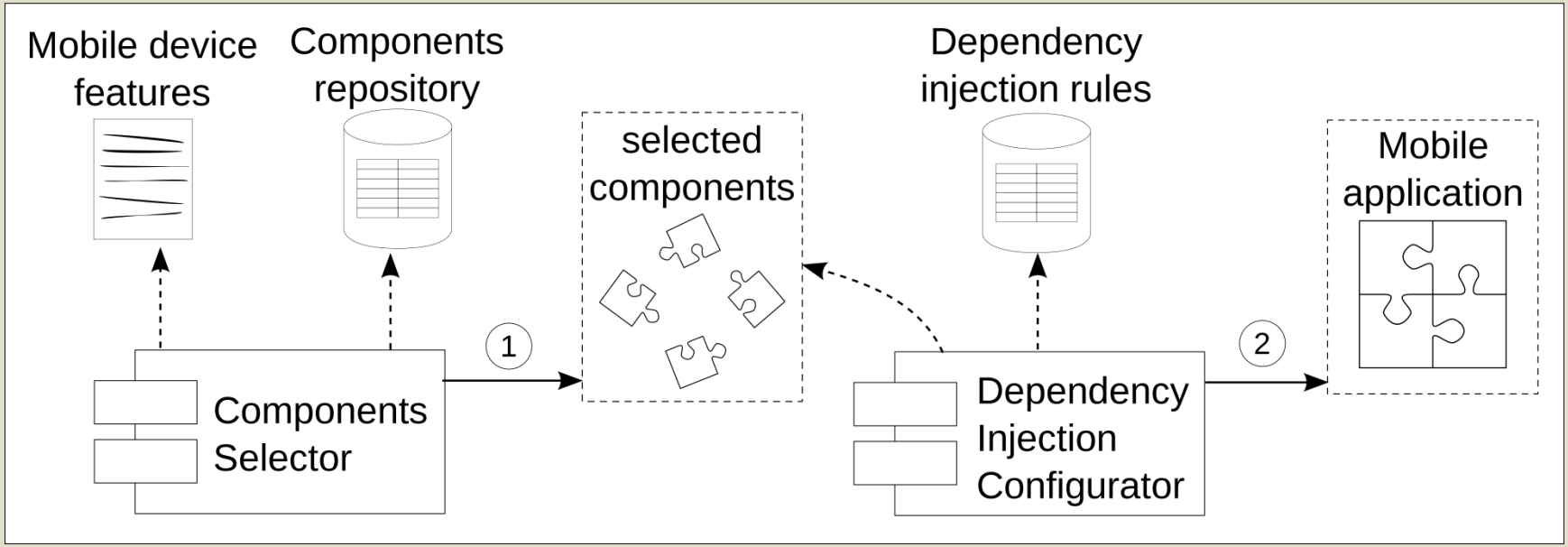
```
<componentData>
    ...
    <component id="03" class="Menu1">
        <constraints>
            <constraint name="screen" value="360x640" />
        </constraints>
    </component>

    <component id="04" class="Menu2">
        <constraints>
            <constraint name="screen" value="240x320" />
        </constraints>
    </component>
    ...
```

# Our Solution

1. Capture of the device features.
2. Selection of the software components that are compatible with the captured features.
3. Configuration of the component dependencies.
4. Building of the mobile application.

# Our Solution

- The selection of the software components that are compatible with a specific mobile device.



# Why is it interesting?

- Provides portability through the use of software components.
- Dynamic and automated composition of mobile applications.
- The developers are free from the responsibility of manually selecting and composing the software components.

# Why is it interesting?

- Building of mobile applications without knowing the device model.
- Key benefits:
  - **Flexibility**: mobile applications can be deployed to different devices.
  - **Reusability**: reuse of the existing software components.
  - **Loose coupling**: one component don't have direct knowledge of another.

# Conclusions

- Automating the selection and composition of software components for building mobile applications is very useful when the SPL has many components.
- AppSpotter can be used to select and to compose of software components for building mobile applications.

# Conclusions

- Achieved results:
  - A mechanism for selection of software components.
  - A data-driven approach to represent the component dependencies.
  - A dependency injection framework to assemble the components together.
  - The possibility of building mobile applications by using the device features.

# Acknowledgments




UFAM



FAPEAM

Vicente Lucena Jr, UFAM

*Questions?*

*Thanks for your attention*!