Incremental, two-level deadlock analysis for incomplete Java Card 3.0 programs

Rebekka Neumann, Michael Thies, Uwe Kastens

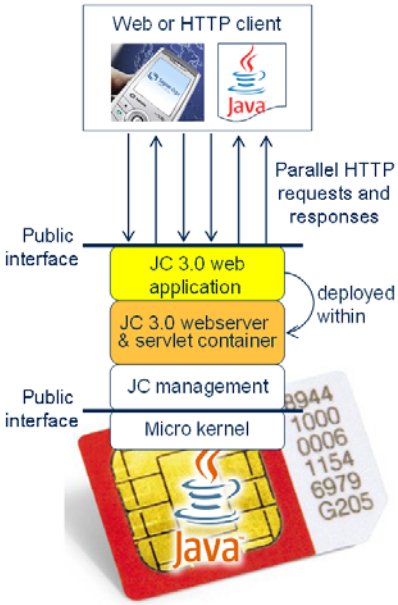
University of Paderborn

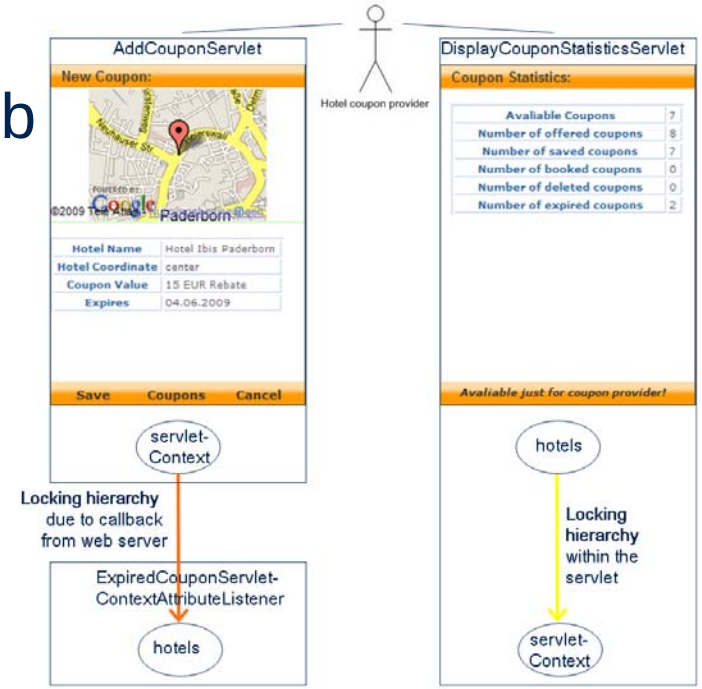
3rd of July 2009

© Software Quality Lab (s-lab)

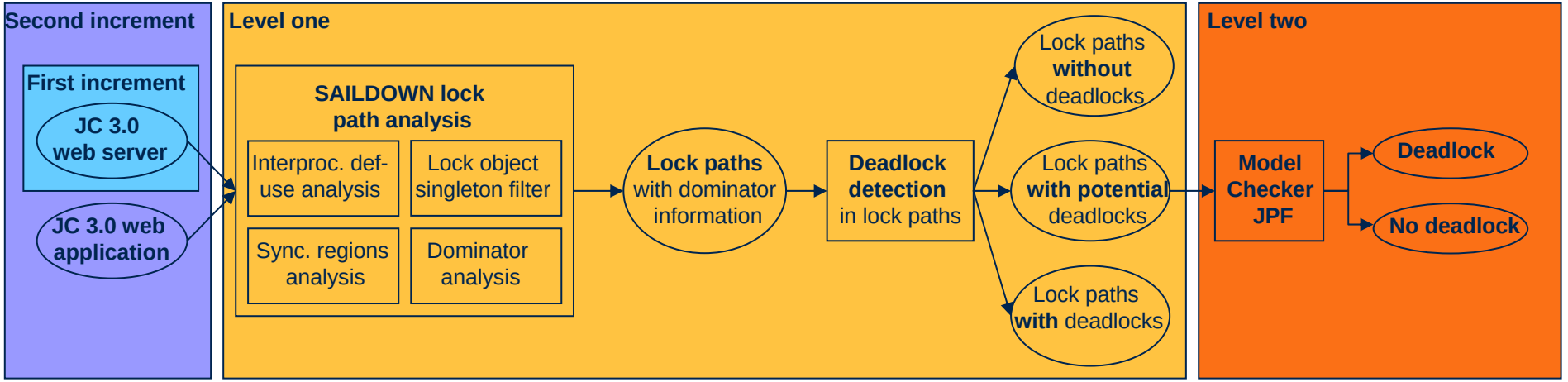
Motivation for deadlock detection in incomplete Java Card (JC) 3.0 programs

- **JC 3.0 smart cards** are an **internet-ready interface**
- **Concurrency** is a **new language feature** for Java on these smart cards
- **Example:** dead-lock in JC 3.0 web application





Incremental, two-level deadlock analysis for incomplete Java Card 3.0 programs



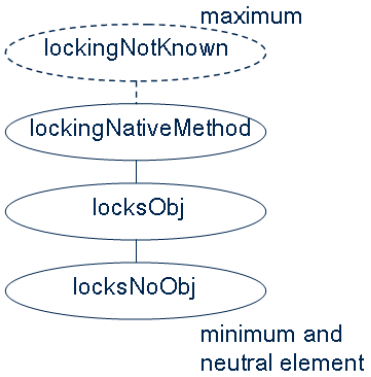
Steps of deadlock analysis:

- Level one, first → second increment:
  - **static program analysis** for incomplete JC 3.0 web server → JC 3.0 web server completed by a JC 3.0 web application
  - Calculation of lock paths for all its fixed methods → all methods
  - Detection of global and local deadlocks → **reuse** of static analysis results
- Level two:
  - **model checking** to check all potential deadlocks
  - **reuse** of static analysis results to guide the model checker and to detect local deadlocks

First evaluation result

- **Goal:** determine the percentage of **reusable static analysis results** for the JC 3.0 web server including the JC 3.0 API

- **Procedure:** conduct a SAILDOWN analysis to calculate the **locking behavior** and the **dependence** of each method from other methods



- **Results:**
  - **~ 50% of the static analysis results can be reused**
  - **~13% of the methods contain locking behavior**, whereof **~25% are fixed** and **~ 75 % are extendable**

Status and perspectives