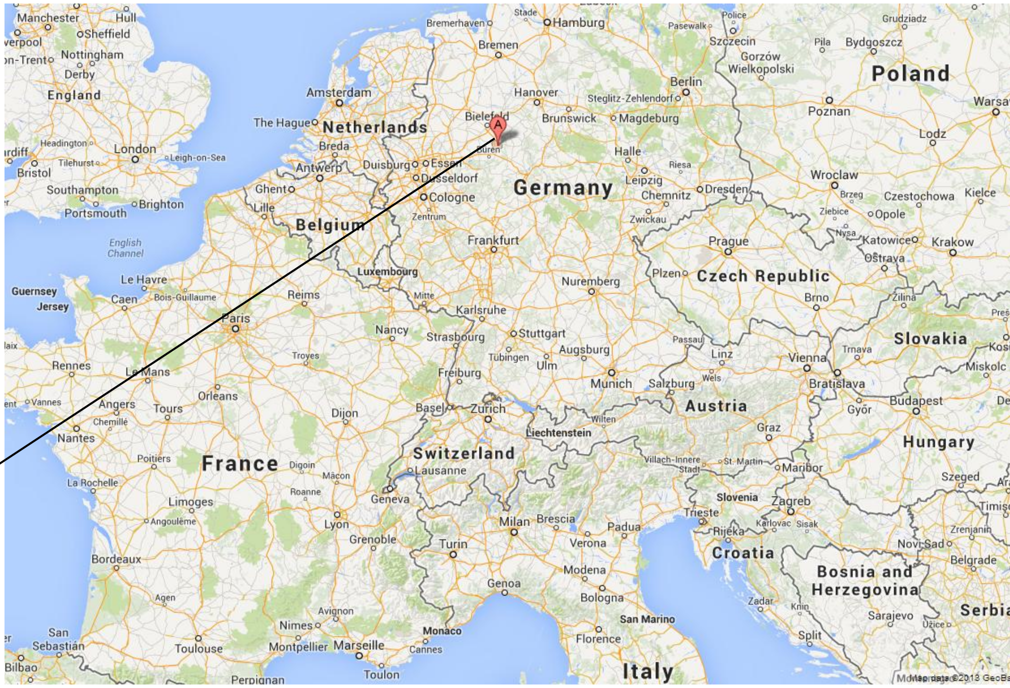
Early Phase Memory Leak Detection in Embedded Software Designs with Virtual Memory Management Model

Mabel Mary Joy
University of Paderborn / C-Lab



https://maps.google.com

Introduction

- Virtual platforms –in early design tests
  - Detect faults early
  - Optimize the design

- Memory limitations
  - Memory is constrained
  - Memory leaks cannot be afforded

- Increasing complexity makes early design tests essential
- Memory leak detection at early stages helps reduce flaws at production
  - But memory leaks are hard to detect

Memory Leak

- Memory leak occurs when a memory location is not freed after it‘s use
- Program run out of memory and eventually fail
- The code at the point of failure often has nothing to do with the leak
- Considered as "hidden" problems as they are hard to detect
- For non-garbage collected soft real time systems an automated and fast leak detection at early stage is required

Memory Leak

A sample code with memory leak

```
char *mem_area =malloc(10)
char *new_area = malloc(10)
memory_area=new_area
```

Outline

- Introduction
- State of the art
- Virtual memory modelling
- Memory leak detection approach
- Conclusion and Future Work

Related Work in Memory Leak Detection

- Different methodologies adopted to reduce leak
- Classification
  - Based on the Problems Addressed
    - Leak detection
    - Leak elimination
    - Leak toleration
  - Based on Methodologies
    - Static
    - Dynamic
- Different tools available
  - Commercial and non-commercial
    - Mtrace, Dmalloc, Memwatch, Valgrind, Purify

Methodologies

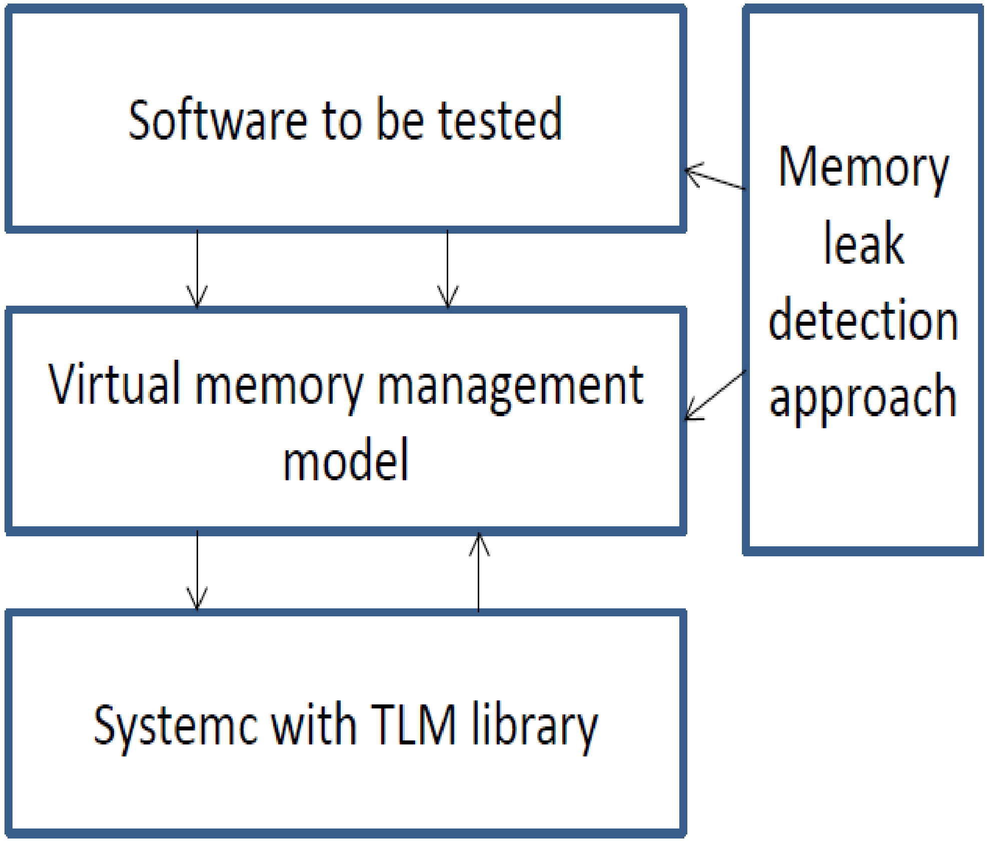
- Static Methods
  - Annotations
  - Ownership models
  - Leak Analysis by contradiction
  - Symbolic pointer analysis
  - Parameterized procedural summaries
  - Flow analysis

Methodologies

- Dynamic Methods-Used at runtime
  - Reference counting
  - Reachability
  - Liveness
  - Combinations of two or more

Our approach

- Memory leak detection with virtual platforms
- Virtual prototyping with SystemC
- Leak detection at simulated abstract levels
- Faster and early analysis


Software to be tested
Virtual memory management model
Systemc with TLM library
Memory leak detection approach

Our approach

- Simulated model implemented with TLM 2.0 library in SystemC
    - TLM library focus on the transaction model
- ARM 9 processor with Integrator CP board is considered for case study

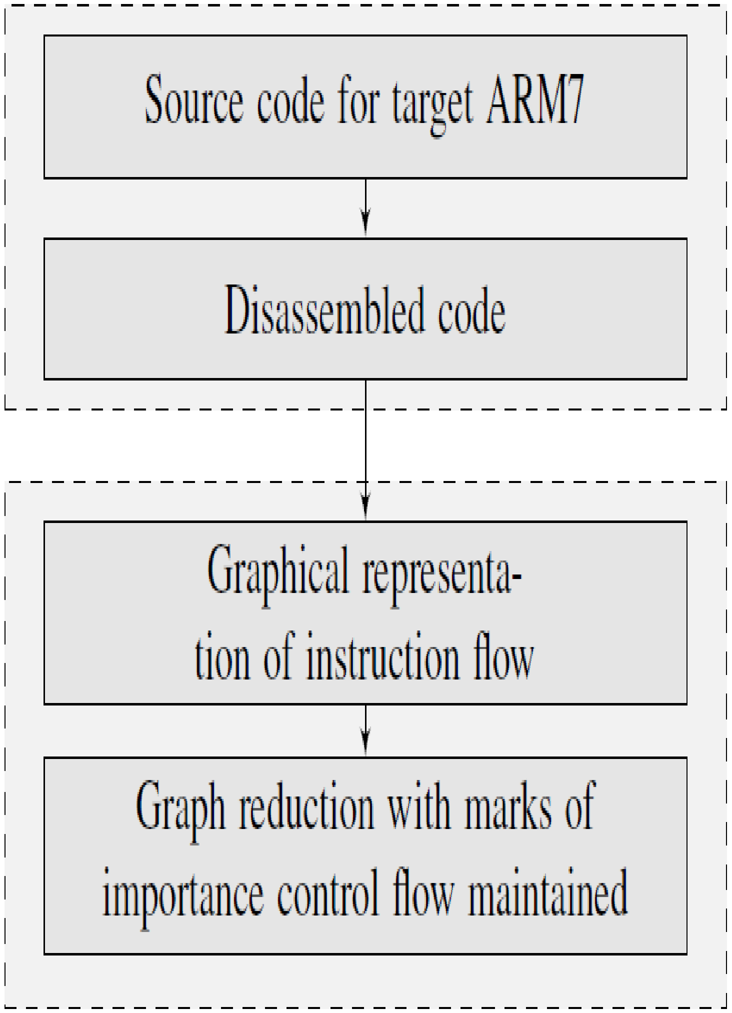
- Currently C and C++ programs for soft real time systems are tested

Virtual memory modelling

- Simulation framework
  - Enables analysis of modules developed independently and integrated later
  - Real target environment is not necessary (which may be expensive)
  - Simulation environments could be reused and customized on case by case
  - Flexible and portable

Virtual memory modelling

- Simulation framework
- Rapid and early feedback of design decisions
- Simulations at higher abstraction levels
- Fast and enough accuracy is required
- Abstraction rules out the technical dependencies of dependent protocols
- E.g. ARTOS framework, with various API to run the software at abstracted levels


Source code for target ARM7
Disassembled code
Graphical representa-tion of instruction flow
Graph reduction with marks of importance control flow maintained

Memory leak detection

- Hybrid model over the simulated framework
  - Software undergoes a combination of static and dynamic analysis for leak detection
- Static analysis involves automated control flow graph generation techniques
- Path search is done to detect breaks in the allocate free graph
- A one to one mapping is generated for each allocate-free combination

Memory leak detection

- Hybrid model over the simulated framework
  - Software undergoes a combination of static and dynamic analysis for leak detection
- Dynamic analysis
  - Static analysis fails to detect all possible leaks, such as dynamic references
  - The memory arbiter in the simulated model, will be utilised to simulate the request –allocation.
  - Approaches of reference counting and liveness analysis combined with offline analysis is used

Conclusion

- Memory leaks are serious problems
- Detecting them at early stages is very important, especially when there is no explicit garbage collector
- We propose a novel hybrid technique at simulated level
- Simulated model at abstracted levels in TLM
    - Makes testing faster and focused on the testing entity
    - Flexible and customizable
    - With enough speed and accuracy

Future work

- The proposed approach is at an early implementation stages
  - Need full implementation
- Case studies and analysis with comparison to existing state of the art approaches are yet to be done
- Memory model could be extended to detect other memory related problems like fragmentation and corruption