Efficient Instrumentation For Performance Profiling

Edu Metz, Raimondas Lencevicius

Nokia Research Center

Software Architecture Group, Boston

5 Wayside Road, Burlington, MA 01803, USA

E-mail: Edu.Metz@nokia.com,
Raimondas.Lencevicius@nokia.com

NOKIA

Motivation

- Performance profiling involves tracing software execution and analyzing obtained traces
- Traces affect system performance and distort software system execution
- Need to minimize effect of tracing on system's performance
- Trace set needs to be optimized according to performance profiling problem being solved

Position

Minimization of the effect of tracing on the underlying system's performance can be achieved only by adding the software trace design and implementation to the overall software development process

Overview

- Performance profiling
- Efficient instrumentation
- Process for efficient trace instrumentation
- Example
- Why it will fail
- Why it will succeed

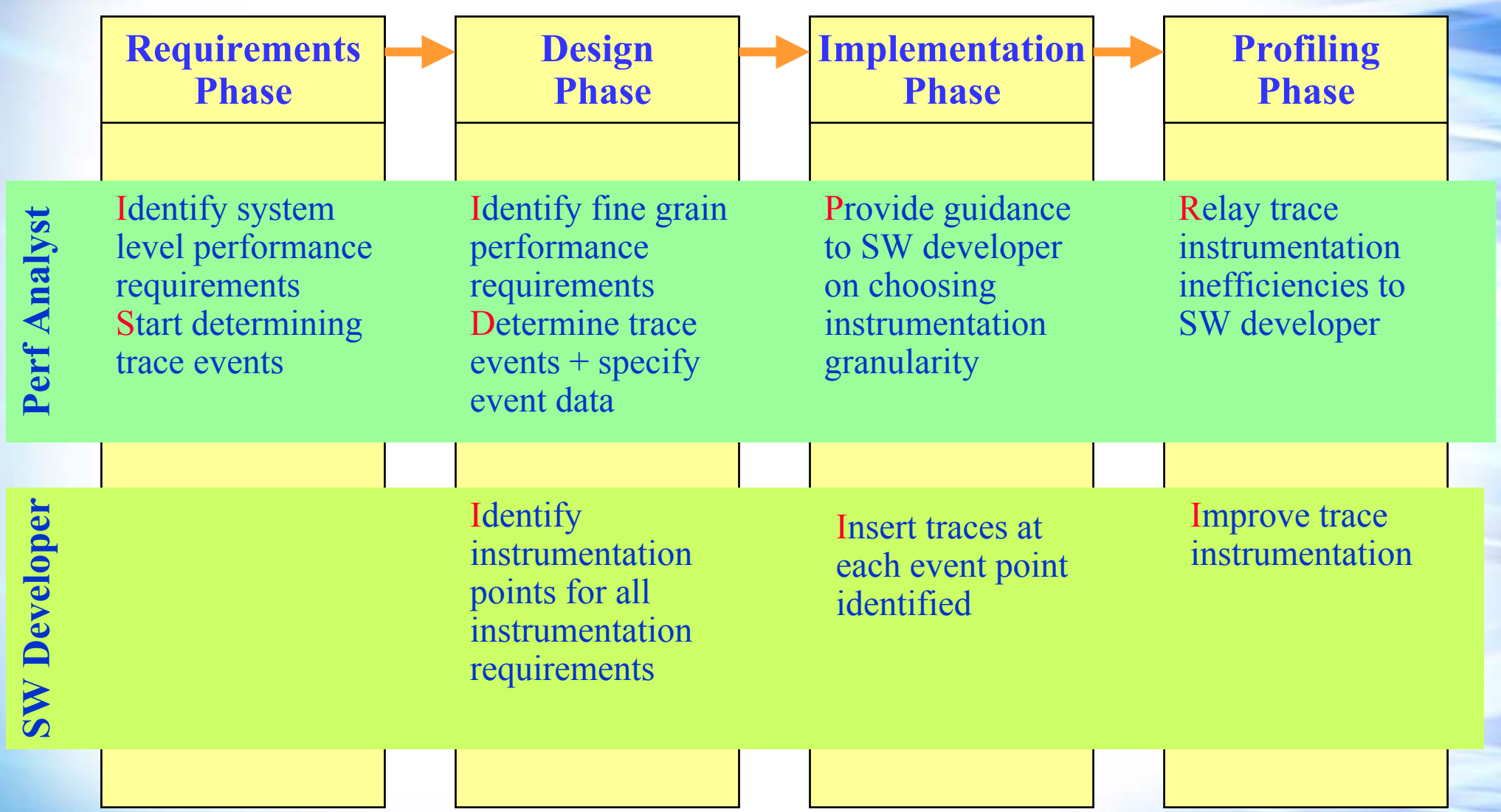
Performance Profiling

- Determine performance parameters of a running software system
- Various types of event information can be obtained
  - component entry and exit
  - function calls
  - execution states
  - message communication
  - resource usage
- Tracing reduces validity of performance profiling
- Trace instrumentation comes at a cost
  - takes time
  - changes behavior of software system
  - could violate real-time constraints and timing requirements in real-time systems
- Need to minimize performance impact of trace instrumentation

Efficient Instrumentation

- Requirements
  - minimize number of instrumentation points
  - minimize runtime overhead
  - guarantee constraints and requirements
- Meeting these requirements can be a complicated task
  - software development and performance profiling often performed by different individuals
  - software developers and performance analysts have different knowledge and skill sets
- *Proposal: Draw upon knowledge and skills software developers and performance analysts bring with them and use this knowledge to create efficient trace instrumentation*

Efficient Trace Instrumentation Process

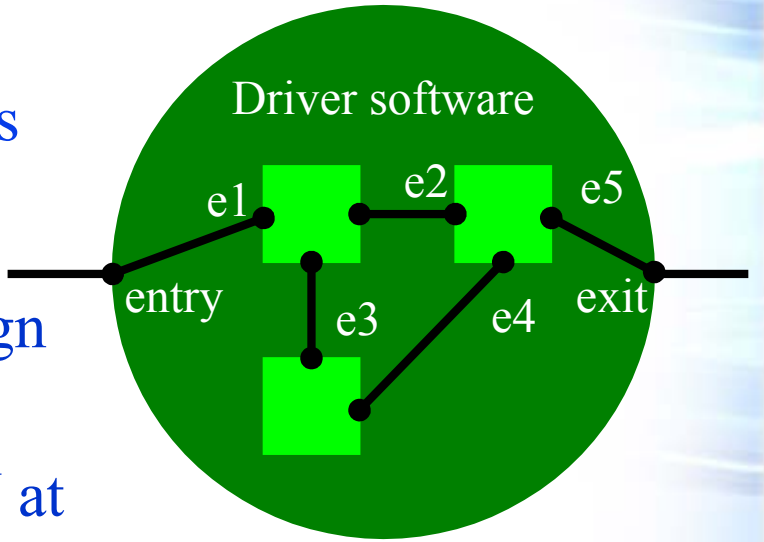


Example

- Energy is an important performance requirement in mobile devices
  - consumption varies depending on HW resources used
- During execution software access HW resources
  - need to monitor HW accesses to determine when a HW resource is used
- Which HW access events should be traced?
  - trace all access events or just enable and disable events?
  - best answered by performance analyst

Example (cont.)

- Requirements phase
  - performance analyst identifies power consumption requirements of HW resources
- Design phase:
  - performance analyst identifies HW access events to be traced
  - SW developer identifies corresponding instrumentation points in driver SW design
- Implementation phase
  - SW developer inserts traces in driver SW at each event point identified
  - performance analyst provides guidance to SW developer on choosing instrumentation granularity



NOKIA

Why it will Fail

- Good follow through by both performance analyst and software developer is essential. If one slacks off, the entire effort suffers
- Benefits only visible to performance analyst
- Developers may not be too eager to adapt
  - increases design and implementation workload
  - may not have adequate time
  - insert event traces but they will not use them
  - developers don’t like change
  - most developers are ‘trained’ to wait for hardware advances to provide performance improvements

Why it will Succeed

- Adding trace instrumentation for performance profiling to software development process:
  - has potential to decrease number of trace instrumentation points
  - would not trace more event data then needed to profile performance
  - would reduce impact of trace instrumentation on software system performance
  - allows for creating ‘standardized’ performance trace instrumentation
  - provides formatting rules for performance event data

Questions ?