SAAT: Reverse Engineering for Performance Analysis

2004.05.25.

Seon-Ah Lee, Seung-Mo Cho, Sung-Kwan Heo
Mobile Solution Group
Software Center
Samsung Electronics
salee@samsung.com

Agenda

I. Introduction & Background

II. Concept of SAAT

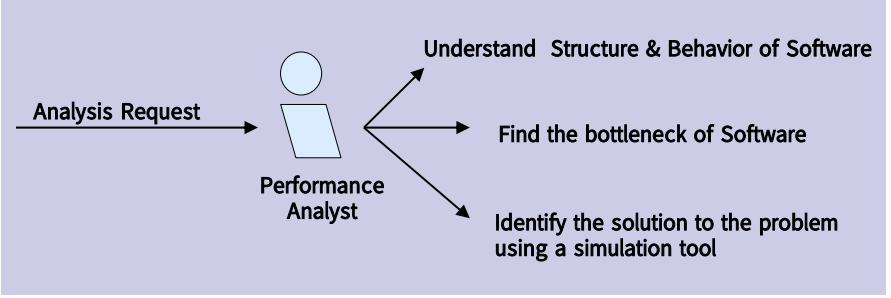
III. Issues & Conclusion

I. Introduction & Background

- Motivation
- Challenge
- Software Performance Model
- Software Reverse Engineering

Motivation

- **For performance analysis at our research center**
  - **We hoped to provide the result of performance analysis as quickly as possible to stakeholders**



  - **We decided to automate the process in order to shorten the time of the performance analysis**

Challenge

- **There is no tool showing software execution structure with performance information**

- Performance Analysis Tool ≠> Software Behavior
  - Tools such as gprof, Vtune and Quantify show call graphs and bottleneck candidates, but does not show the execution flow, so we cannot use the call graphs for understanding system's behavior.
- Reverse Engineering Tool ≠> Performance information
  - Static analysis tools such as Source Insight and aiCall show a software structure, but it is different from a dynamic structure.
  - Imagix show both static structure and call graphs, but it is only the combination of upper tools.

Software Performance Model

- **Research trends**
  - Researchers in performance engineering are studying how to integrate software architecture with performance information
- **The representative works**
  - C.U. Smith and L.G. Williams, *Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software,* Addison-Wesley, Sept. 2001.
  - Performance model & PASA methodology: extracting architecture information from developer interviews and work products

Developers are always busy

- **Our position**
  - Automation: the interview task ➔ An analysis tool

Software Reverse Engineering

- **Research Trends**
  - Dynamic reverse engineering to extract software execution models from existing systems is also being tried
- **The representative works**
  - T. Systa, "Understanding the Behavior of Java Programs", *In Proc. of the 7th Working Conference on Reverse Engineering,* pp. 214-223, Brisbane, Australia, November 2000.
  - Shimba tool: producing sequence diagrams using trace information at runtime (Java)

No Structure & Performance Information

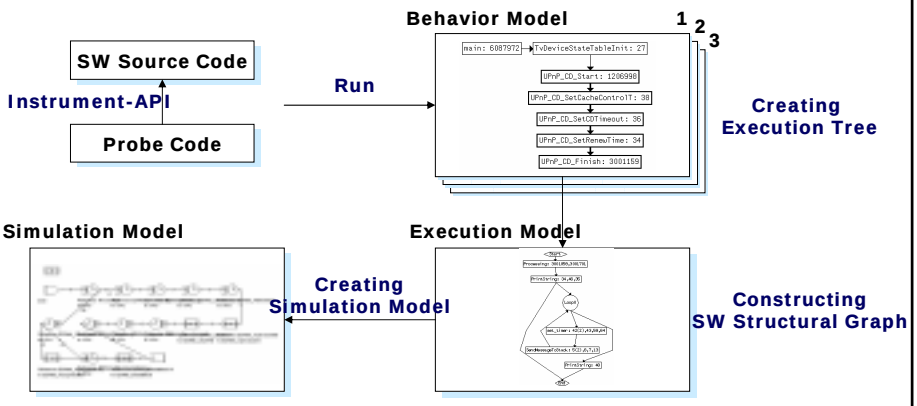
- **Our position**
  - Simulation Model: Execution Structure + Performance Information
  - Suitability: Java ➔ C

II. Concept of SAAT

- What is SAAT?
- Trace Data
- Behavior Model
- Execution Model
- Simulation Model
- Structure of SAAT
- Using this tool in performance analysis

What is SAAT?

- **SAAT**
  - **A tool generating a simulation model from source code automatically**



Trace Data

- **A record of the interaction between the software modules (Using TAU, generating Software Trace Data)**

```
Function Main(parameter seq)
{
    call A(seq);
    call B();
    call C();
}
Function A(parameter order)
{
    if (order = first_sequence)
    {
        call a();
        call b();
    }
    else
    {
        call c();
    }
}
Function B()
{}
Function C()
{}
```

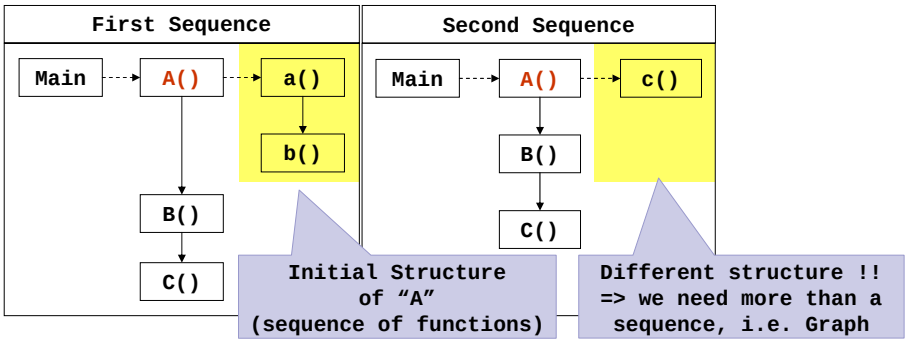
```
A node  Time      Start/Finish

Main    start  time of Main  start  flag
  A     start  time of A     start  flag
    a   start  time of a     start  flag
    a   finish time of a     finish flag
    b   start  time of b     start  flag
    b   finish time of b     finish flag
  A     finish time of A     finish flag
  B     start  time of B     start  flag
  B     finish time of B     finish flag
  C     start  time of C     start  flag
  C     finish time of C     finish flag
Main    finish time of Main  finish flag
```

Begin Point of A

End Point of A

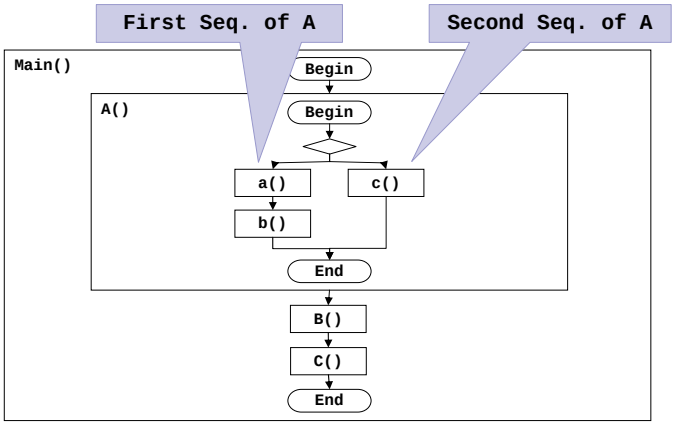
Behavior Model

- A diagram representing Software Trace Data as nodes and edges
  - Node: a function
  - An edge directing left: call-relationship between two functions
  - An edge directing below: execution order between two functions



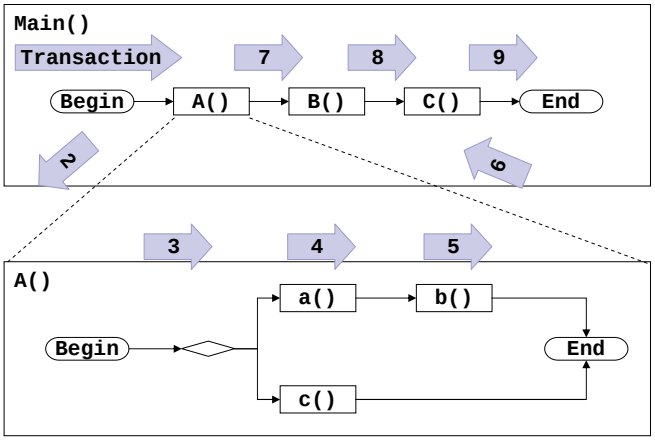
Execution Model

- A composite of several Behavior Models to represent the dynamic structure of the software



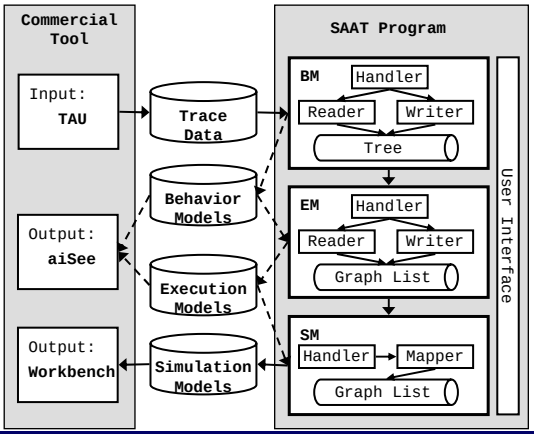
Simulation Model

- A model running in a simulation environment to demonstrate architectural issues of the present system.



Structure of SAAT

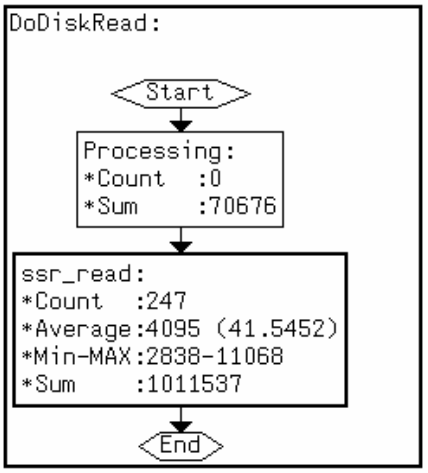
- SAAT is related to existing commercial tools.
  - TAU: Generating Trace Data
  - aiSee: Presenting Trees and Graphs
  - Workbench: Providing Simulation Environment



- BM
  - Creating a binary tree representing one execution of software
- EM
  - Creating a graph for representing software run-time structure
- SM
  - Creating a model running simulation environment.

Using this tool in performance analysis

- **We can insert the statistic data into each node from execution information of Trace Data**



III. Issues & Conclusion

- Why to drop static analysis
- How to construct execution model from behavior model
- Fail or Succeed
- Contribution
- Future Works

Why to drop static analysis

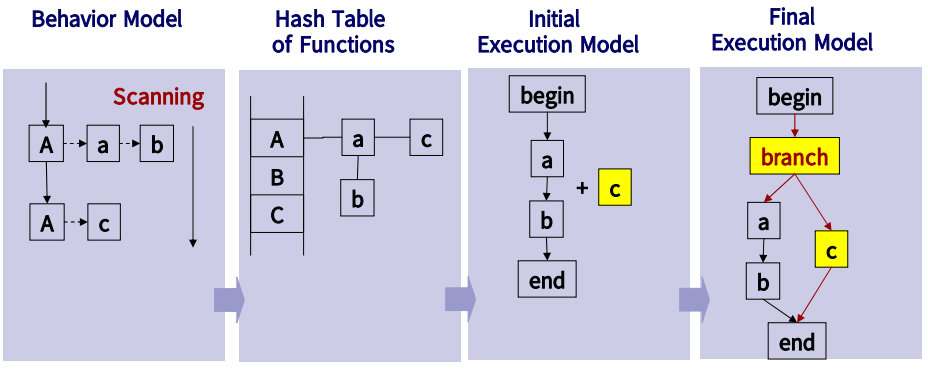
- Extracting more precise structural information of existing software
- Including all structural information for analyzing the software

- Needing Performance information
- Simplifying the structure of software in our focal aspect
- reducing the complex manipulation of our tool

How to construct execution model from behavior model

- Conditions
  - A function with two or more invocation sequences
  - the sub-sequences of A function are different

➔ the function should have loops or branches



Fail or Succeed

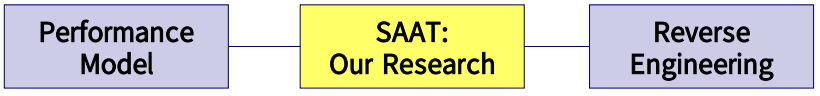
SAMSUNG

- This tool might fail?
  - The trace and behavior models can get large, so the computation of an execution model gets complex
  - The result to generate simulation model is rough, so user mediate part should be appended to the tool
- This tool might succeed?
  - This tool was used to discover the pattern of software execution and useful to find problematic part already
  - This tool is the first trial to extract simulation model from the existing software automatically

Contribution

SAMSUNG

- Benefits
  - Acquiring the methodology facilitating the understanding of existing software
  - Developing a suitable tool to analyze our embedded software implemented in c language
  - Pioneering the way to create a model running a simulation environment from existing software automatically
- Academic interest
  - Bridging the gap between performance analysis and reverse engineering.



Future Works

SAMSUNG

- **Grouping functions**
  - **The way to bind the corresponding component**
  - **User intervention parts / Component-declaring parts**
  - ***Precedence: Dali Workbench tool made by Kazman***
- **More refined Models**
  - **Additional rules for converting Behavior Models to an Execution Model**
  - **Options to modify the Simulation model for user tastes**
- **Various Model Environments**
  - **Adapting this tool to several modeling environments.**