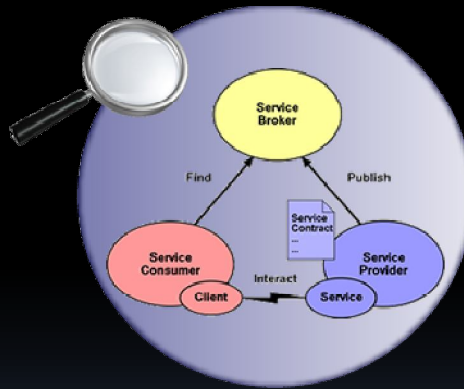




TOWARDS SOA MONITORING



PhD. Ionuț Apetrei, FII, UAIC
Iași – România



Agenda

- 1. Introduction
- 2. Analyzing SOA
- 3. The need for monitoring
- 4. SOA Monitoring group at FII, UAIC
- 5. Conclusions

1. Introduction

- The great expansion of software systems in the last decades, emphasized the need of creating robust systems that run in a stable environment (in other words specification is met in the implementation and execution)
- Our quest has a well defined goal, searching and proposing solutions for runtime verification of distributed software systems

2. Analysing SOA

- As mentioned earlier we are interested in observing and verifying complex software systems
- The current software structure paradigm taken into consideration is SOA (Service Oriented Architecture), which is included in the higher perspective, known as SaaS (Software as a Service)
- We will define the service concept from a SOA perspective

2. Analysing SOA - Basics

- The building block of SOA paradigm is the service concept, it's properties are:
 - defined by an interface (can be platform independent)
 - accessible through a network
 - it's operations are defined at the interface level
 - it's interface and implementation can be decorated with extensions (ex. WS-ReliableMessaging etc.) that will be taken into considerations at runtime

3. The need for monitoring

- We will start by laying out the general context
- The idea of monitoring is closely related with the one of verification
- Three main approaches regarding verification must come in our minds, when speaking of it:
 - **Theorem proving** – targets the demonstration of program correctness using mathematical tools
 - **Model checking** - automated verification method, requires systems with finite states. It uses temporal logics.
 - **Testing** - is defined by the process of finding flaws in a system

3. The need for monitoring

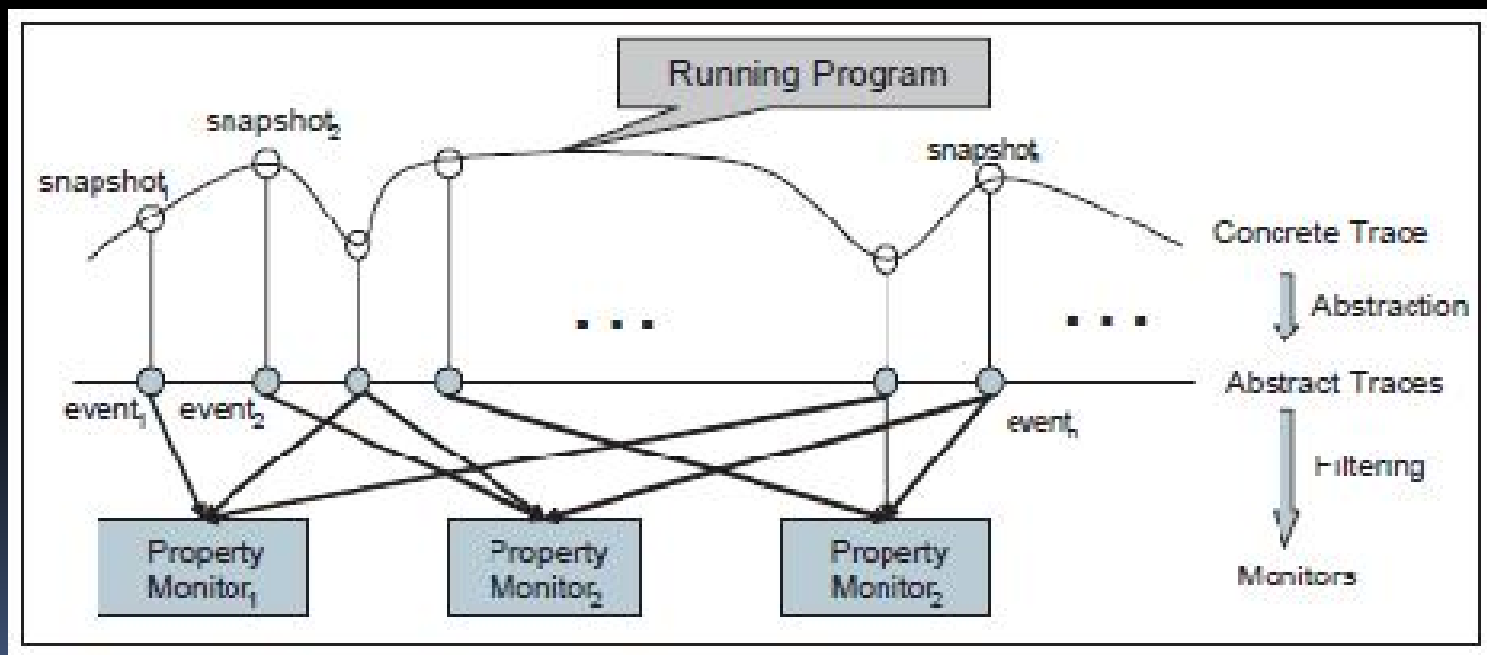
- The above approaches are known as **heavyweight techniques**, encountered in static analysis.
- **Runtime verification** is a **lightweight techniques**, it targets only the detection regarding the violation or validation of correctness properties.
- A core concept in RV is the idea of **monitor**. It's task is to decide if the **current execution** satisfies a certain property.

3. The need for monitoring

- RV in the SOA context
 - Identifying the problem: SOA is a software architecture paradigm that insists on distributed systems. Much effort was made in order to create standards and software infrastructures. The next step is the development of techniques and technologies which will enable software services to adapt (take correction measures) based on runtime verification.
 - Formulating the solution: use techniques specific to RV, integrate the MOP paradigm (Monitor Oriented Programming – developed by prof. Grigore Rosu and its team at University of Illinois)

3. The need for monitoring - Monitoring-Oriented Programming

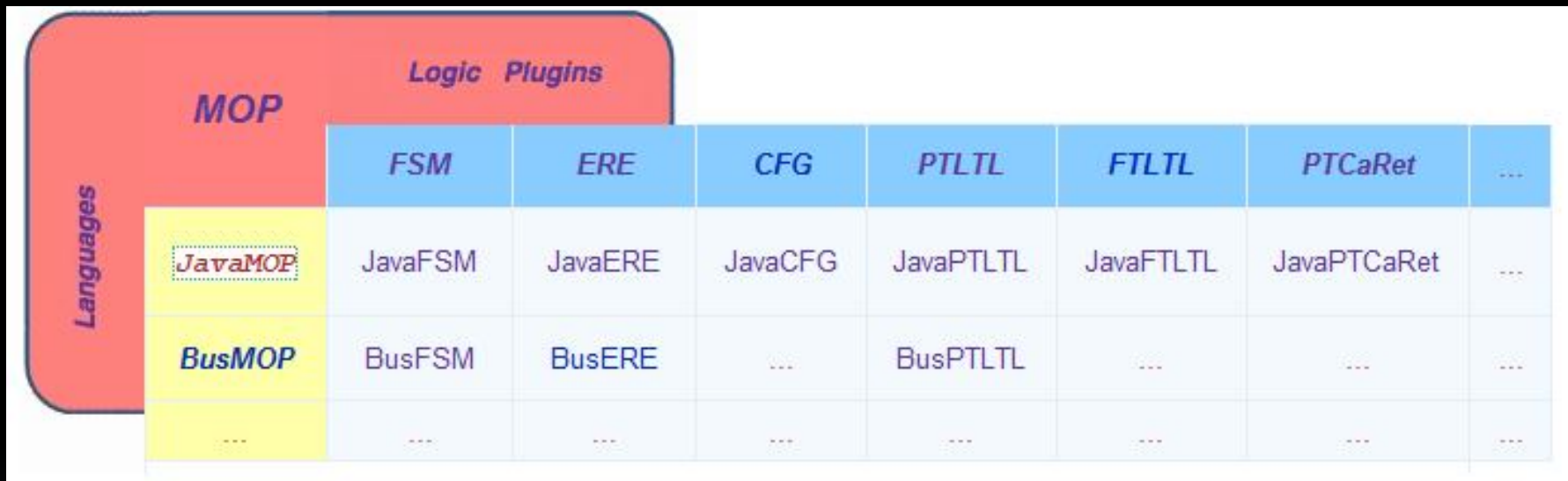
- Below we present the MOP monitoring model



© FSL Laboratories, UIUC

The monitor structure from MOP perspective - $M=(S, \epsilon, C, i, \sigma, \gamma)$

3. The need for monitoring - Java MOP



© FSL Laboratories, UIUC

The above formalisms are meant to be used in order to specify the system properties, that will be monitored MOP, in instance Java-MOP excels because it allows runtime modification of a system, and it does not constrain for using a certain property formalism

LTL operators --- ftLTL : $\circ F$ (next) $\diamond F$ (eventually) $\square F$ (always) $F U F'$ (Untill)

--- ptLTL: $\bullet F$ (previous) $\blacksquare F$ (always) $\diamond F$ (eventually) $F S F'$ (Since)

3. The need for monitoring – JavaMOP – Example

```
package mop;

import java.io.*;
import java.util.*;

HasNext(Iterator i) {
    event hasNext after(Iterator i) :
    call(* Iterator.hasNext()) && target(i) {}
    event next before(Iterator i) :
    call(* Iterator.next()) && target(i) {}

    ptl tl : next implies (*) hasNext

    @violation{
        System.out.println("next called without hasNext!");
    }
}
```

----- monitor specification

```
Vector v=new Vector();
v.add(new Integer(10));
Iterator i =v.iterator();
v.add(new Integer(20));
System.out.println(i.next());
```

JVM: ConcurrentModificationException

In the left side we can see the monitor specification for the **fail-fast** property of an iterator in Java.

4. SOA Monitoring Group at FII, UAI C

- It's main purpose is to implement RV techniques and to integrate MOP paradigm into SOA technologies.
 - Current technologies on which we work:
 - Web services – JAX-WS (JBoss Native Stack)
 - Business processes- jBPM 5.0
 - Software components – EJB Framework

4. SOA Monitoring Group - WS

- Our solution follows these steps:
 - use a WSDL file
 - generate the Java web-service implementation (top-down approach)
 - specify the system properties
 - generate the monitoring code based on system properties
 - attach the monitor to the web-service (takes place at the application server level)
 - execute the entire system

4. SOA Monitoring Group - BPs

- We are currently developing an Eclipse Plug-in for real-time monitoring of business processes
- The current specification considered is BPMN 2.0
- We focus on Java enterprise software infrastructure - in particular jBoss application server
 - jBoss Governer
 - jBoss Drools

4. SOA Monitoring Group - EJB

- We have created a tool which is capable of intercepting EJB software components and test them.
 - EJB Stateless Bean
 - EJB Statefull Bean
 - EJB Message Driven Bean
- The next step will target dynamic intercepting deployed software components
- We are also interested in applying exogenous coordination models

EJB Interceptors Benchmark

Insert title here - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://localhost:8080/StatisticsInterface/Faces/listStatistics.jsp


Google

Statistics:

EJB name	EJB type	EJB event	Start time	End time	Success	Parameters	Result
PersonBean	STATELESS	afisare2	2010-12-02 19:56:04.296	2010-12-02 19:56:04.359	true	No parameters	Void return value
PersonBean	STATELESS	insert	2010-12-02 19:56:16.953	2010-12-02 19:56:17.625	true	John, Doe	Void return value
MyMDB	MESSAGE_DRIVEN	onMessage	2010-12-02 19:56:17.703	2010-12-02 19:56:17.703	true	message	Void return value
EmployeeBean	STATEFUL	afisare2	2010-12-02 19:56:40.718	2010-12-02 19:56:40.718	true	No parameters	Void return value
EmployeeBean	STATEFUL	insert	2010-12-02 19:56:40.765	2010-12-02 19:56:40.765	true	Johny, B	Void return value
EmployeeBean	STATEFUL	afisare2	2010-12-02 20:03:00.156	2010-12-02 20:03:00.187	true	No parameters	Void return value
EmployeeBean	STATEFUL	insert	2010-12-02 20:03:00.234	2010-12-02 20:03:00.25	true	Johny	Void return value
MyMDB	MESSAGE_DRIVEN	onMessage	2010-12-03 16:40:07.578	2010-12-03 16:40:07.578	true	message	Void return value
EmployeeBean	STATEFUL	afisare2	2010-12-03 12:24:53.765	2010-12-03 12:24:53.859	true	No parameters	Void return value
EmployeeBean	STATEFUL	insert	2010-12-03 12:24:53.953	2010-12-03 12:24:53.968	true	Test1, Test2	Void return value
PersonBean	STATELESS	afisare2	2010-12-03 12:25:37.64	2010-12-03 12:25:37.64	true	No parameters	Void return value
PersonBean	STATELESS	insert	2010-12-03 12:25:37.656	2010-12-03 12:25:37.843	true	Test3, Test4	Void return value
MyMDB	MESSAGE_DRIVEN	onMessage	2010-12-03 12:25:37.859	2010-12-03 12:25:37.859	true	message	Void return value
EmployeeBean	STATEFUL	afisare2	2010-12-03 16:36:27.843	2010-12-03 16:36:27.843	true	No parameters	Void return value
EmployeeBean	STATEFUL	insert	2010-12-03 16:36:27.937	2010-12-03 16:36:27.937	true	,	Void return value
PersonBean	STATELESS	afisare2	2010-12-03 16:40:07.484	2010-12-03 16:40:07.5	true	No parameters	Void return value
PersonBean	STATELESS	insert	2010-12-03 16:40:07.5	2010-12-03 16:40:07.578	true	,	Void return value



5. Conclusions

- We focus on analysing and extend verification formal methods
 - We are developing a RV verification framework which will be capable of monitoring SOA specific technologies: Web Services, Business Processes and software components (EJB)
- 



The end

- Questions?
- 