Osmius 9.07
The OpenSource Monitoring Tool

You can't manage what you can't see.
What is Osmius?

Osmius allows you to **monitor** and keep track of **everything** connected to a **network**.
What is Osmius?

We've designed Osmius to monitor:
Computers and Network devices
Applications and Databases
User transactions...

But also:
Industrial sensors.
Energy consumptions.
Pollution levels in the neighbour.
Stock shares.
Any idea?
Why choose Osmius?

Briefly, with Osmius you can:

Monitor **thousands of devices** in Real Time.
Align IT with the **Business** (SLA Management).
**Data Mining** and Business Intelligence.
Use the power of **GIS** to see/analyze data.
Monitor new “things” in minutes.
Send automatic **notifications** to the staff.
**Predict Capacity** shortages (ITIL).
Access ALL the **code and documentation**.
Who is behind Osmius?

Investments

Peopleware
Funds to Help R&D

Technological Partners

Geographic Systems
Business Intelligence
Multiplatform Installers

Osmius Partners

Installation  Deployment  Support
Administration  Development  Training
Community

In Osmius everything is Open... not just the “core” !!

First Stable version released in July 2008
More than 4,000 downloads
SourceForgeRanking : ~ Top 50-100
Central Server
Process to receive and **correlate** events and to manage all the agents

DataBase
Open and documented DataModel. Events, SLA, etc

WebConsole
J2EE connecting to the DBase.

**Agent**
Specialized on monitoring specific instances like MySQL, Web sites...

**Master Agent**
Point from which you want to monitor and deploy Agents. You can use 1 to N

**Instances**
Something you want to monitor (Web, DB, Temp)
Central Server
C++ & ACE & MySql API
Fast insert and Correlation
Secure Comms

DataBase
Normalized
Explain everything
Stored Procs

WebConsole
Java J2EE
Springs – Hibernate
OpenLayers

Agent
C++ & ACE
API (MySql, Curl, etc)
Only new code

Master Agent
C++ & ACE
Multiplatform
Persistent Queues
Fault Tolerant

Instances
Use API
Reuse Connections

Architecture & Tecnology
Concepts: Easy to understand

**Instance:**
Everything you want to monitor.

**Instance Type or Class:**
Defines the kind of questions you can make to the instance
MySQL Database | Windows Server | Exchange

**Instance Events:**
Depending on its type an Instance can receive certain events
MySQL Database: # Users, Free Space, Kbytes Out,...

Parameters: Every 10 seconds – Customized Warning & Critical Threshold

| Event 1 :: Database Connections is: 250 |
| Event 2 :: Uptime is: 8h 30' |
| Event 1 :: Time to load Web is: 15 s |
| Event 1 :: CPU Load is: 80% |
Correlation Engine I:

The Active Events view should be as clean as possible:
- Show only 1 row for repeated events.
- Automatic clean “bad” events when arrives a “good” one.
- If Temperature is OK clean previous CRITICAL events.

Correlation Engine II (*)

Identify dependencies between Instances:
- If this servers fails → DataBase and WebServer also fail.

Manual and automatic identification.
Concepts: Easy to understand

Service:
A group of instances joined to offer a service to some users
Normally represents some functional group into your business

SLA – Service Level Agreement:
Defines objectives to your services.
The Intranet cannot be down more than 99.999 of its time

Service Working Time
Are we using the Intranet 24x7?
Services must accomplish SLAs within their Working Time
Osmius also implements: **Planned DownTimes**

The Intranet...
Will be down this afternoon from 15:00 to 17:00
due to the upgrade of the Oracle Database from 9i to 10g
Don't take this unavailability into account in the SLAs
Concepts: Easy to understand

SLA Management

- Gold
  - 99.999 Up
  - 99.900 Ok
- Silver
  - 99.999 Up
  - 99.900 Ok

Service Level

- CRM
  - 24x7
- e-Com
  - 24x7
- Intranet
  - 24x7
- eMail
  - 24x7

Instance & Technical Level

- Firewall
- ...}
- Win Host
- Intranet DB
- Host Moon
- Exchange Server
- Exch. FrontE

Rules Propagation

Osmius 9.07: Professional Monitoring
July 2009
Manage the Configuration:
Distribute changes from the console to the distributed agents
Distribute or upgrade agents with one click.

Create your own...
Events using your customized scripts and distribute them.
Agents: Calling Peopleware or building them yourself using the Osmius Development Framework.
User Notificacions & Subscriptions

Let Osmius inform you about changes.

Sending Osmius Event to others

Standalone Agents: Integrate the agents directly with e-mail, Nagios, …

Open Tickets automatically using notifications.

Send Exchange Critical Events to another Monitoring system.

Receiving Events from others

Web Services

Command Line Interface
Osmius Advantages

Deal with **thousands** of Instances and millions of events.
Integrates **SLA** management and **ITIL** best practices.
Manage everything from a **Central Point**.
**Business Intelligence** mining tool.
Leverage the power of **GIS**.
Real **Open**: There is no community **downgraded** version.
You can **influence the RoadMap**.
**Well documented**. We care about documentation.
Next Step:
Osmius and Passive Testing

Osmius is supported by:
Ministry of Science and Education
Ministry of Industry and Trade
CDTI
European Funds
What is PasTe?

- PasTe: **Passive Testing** tool.
What is PasTe?

- PasTe: Passive Testing tool.
- A library to perform passive testing approach for timed systems.
What is PasTe?

- PasTe: Passive Testing tool.
- A library to perform passive testing approach for timed systems.
- Integrated into Osmius as an agent.
Passive testing

In testing, there is usually a distinction between two approaches:

- **Active testing.**
  - We can interact with the system (apply tests).
  - We have an assessment of correctness *before* the system is running.

- **Passive testing.**
  - The system is already running.
  - We cannot interact directly with the system.
  - We can observe its traces.

Example: Large systems working 24x7 since this interaction might produce a wrong behavior of the system.
In testing, there is usually a distinction between two approaches:

- **Active testing.**
  - We can interact with the system (apply tests).
  - We have an assessment of correctness before the system is running.

- **Passive testing.**
  - The system is already running.
  - We cannot interact directly with the system.
  - We can observe its traces.

**Example**

Large systems working 24x7 since this interaction might produce a wrong behavior of the system.
**Timed Framework I**

**Specification**

- **State 1**: $i_1$, $\alpha_1$, 30
- **State 2**: $i_2$, $\alpha_2$, 30
- **State 3**: $i_3$, $\alpha_3$, 30
- **State 4**: $i_4$, $\alpha_4$, 25
- **State 5**: $i_5$, $\alpha_5$, 30
- **State 6**: $i_6$, $\alpha_6$, 15
- **State 7**: $i_7$, $\alpha_7$, 80
- **State 8**: $i_8$, $\alpha_8$, 80
- **State 9**: $i_9$, $\alpha_9$, 200

**Transition Rules**:
- $i_1, \alpha_1, 15 \rightarrow i_2, \alpha_2, 30$
- $i_2, \alpha_2, 30 \rightarrow i_3, \alpha_3, 30$
- $i_3, \alpha_3, 30 \rightarrow i_4, \alpha_4, 25$
- $i_4, \alpha_4, 25 \rightarrow i_5, \alpha_5, 30$
- $i_5, \alpha_5, 30 \rightarrow i_6, \alpha_6, 15$
- $i_6, \alpha_6, 15 \rightarrow i_7, \alpha_7, 80$
- $i_7, \alpha_7, 80 \rightarrow i_8, \alpha_8, 80$
- $i_8, \alpha_8, 80 \rightarrow i_9, \alpha_9, 200$
- $i_9, \alpha_9, 200 \rightarrow i_1, \alpha_1, 15$

**Invariants**

1. $\phi_1 = \text{login} / \text{option screen} / [20, 40], + / [0, \infty], \text{disconnection} \mapsto \text{welcome screen} / [10, 20] \triangleright [35, \infty]$
2. $\phi_2 = \text{login} \mapsto \{\text{option screen}, \text{error user}\} / [10, 40] \triangleright [10, 40]$
3. $\phi_3 = \text{data} / \text{profile screen} / [10, 20], \text{save} \mapsto \{\text{option screen}\} / [20, 40] \triangleright [35, 50]$
4. $\phi_4 = ? / \text{option screen} / [5, 35], \text{marks} \mapsto \{\text{marks screen}\} / [20, 40] \triangleright [30, 70]$

The PASsive TEsting tool and Osmius

Las Navas del Marqués, July 2009
Implementation

Traces
Extension of the classical finite state machine model.

The main difference with FSMs is the addition of time.

\[ M = (S, \mathcal{I}, \mathcal{O}, Tr, s_{in}) \]

- \( S \) is a finite set of states
- \( \mathcal{I} \) is the set of input actions
- \( \mathcal{O} \) is the set of output actions
- \( Tr \) is the set of transitions
- \( s_{in} \) is the initial state
Transition: $tr = (s, s', i, o, t)$

- $s, s' \in S$ are the initial and final states
- $i \in I$ is the input action
- $o \in O$ is the output action
- $t \in \mathbb{R}_+$ is the time that the transition needs to be completed
Specification of SSadmin

\[ i_1 = \text{connect} \]
\[ i_2 = \text{login} \]
\[ i_3 = \text{disconnect} \]
\[ i_4 = \text{profile} \]
\[ i_5 = \text{data} \]
\[ i_6 = \text{cancel} \]
\[ i_7 = \text{save} \]
\[ i_8 = \text{marks} \]
\[ i_9 = \text{register} \]
\[ i_{10} = \text{data\_subject} \]
\[ i_{11} = \text{save\_registration} \]
\[ o_1 = \text{welcome\_screen} \]
\[ o_2 = \text{option\_screen} \]
\[ o_3 = \text{error\_user} \]
\[ o_4 = \text{profile\_screen} \]
\[ o_5 = \text{marks\_screen} \]
\[ o_6 = \text{register\_screen} \]
\[ o_7 = \text{confirm\_screen} \]
\[ o_8 = \text{no\_confirmation\_screen} \]
\[ o_9 = \text{not\_spected} \]
Usual interaction between a student and SSadmin

Welcome page

- The student wants to connect to the system.
- She connects to the web page and the system shows the welcome screen.

\[ i_1 = \text{connect} \]
\[ o_1 = \text{welcome\_screen} \]
Usual interaction between a student and SSadmin II

Logging into the system

When the student sees the welcome_screen she can login into the system. If an erroneous login is introduced then the system shows the error_user.

\[ i_2 = \text{login} \]
\[ o_2 = \text{option_screen} \]
\[ o_3 = \text{error_user} \]
Usual interaction between a student and SSadmin III

**Interaction**

- If the student introduces a correct login, then the system will show the `option_screen` (in 30 seconds).
- If the student wants to disconnect, then she only has to press `disconnection` and SSadmin returns to the `welcome_screen` (in 15 seconds).

**Diagram:**

```
   s0 --i2, o3, 30-> s1
      |           |  
      v           v  
     i1, o1, 15 <- s0

   s1 --i3, o1, 15-> s2
      |           |  
      v           v  
     i2, o2, 30 <- s2

   s2 --i7, o2, 30-> s3
      |           |  
      v           v  
     i9, o6, 200 <- s3

   s3 --i4, o4, 25-> s2
      |           |  
      v           v  
     i8, o5, 30 <- s2

   s2 --i8, o5, 30-> s5
   s3 --i9, o6, 200-> s5
```

- $i_2 =$ login
- $i_3 =$ disconnection
- $o_1 =$ `welcome_screen`
- $o_2 =$ `option_screen`
Each time that a user applies $a$ and observes $z$ and the time required by the transition verifies the restriction $\hat{p}$.

**Formal Definition of Invariants**

- $\phi$ is defined according to the following EBNF:

$$
\phi ::= a/z/\hat{p}, \phi |$

$$
\hat{p}, \hat{t} \in \mathcal{IR}, \; i \in \mathcal{I}, \; a \in \mathcal{I} \cup \{?\}, \; z \in \mathcal{O} \cup \{?\}, \text{ and } \mathcal{O} \subseteq \mathcal{O}.
$$
Specifying properties: Invariants

- Each time that a user applies $a$ and observes $z$ and the time required by the transition verifies the restriction $\hat{p}$.
- Some actions are performed and the time required verifies the time restriction $\hat{p}$.

Formal Definition of Invariants

- $\phi$ is defined according to the following EBNF:

$$
\phi ::= a / z / \hat{p}, \phi \mid \star / \hat{p}, \phi' \\
\phi' ::= i / z / \hat{p}, \phi \mid i \mapsto O / \hat{p} \triangleright \hat{t}
$$

$\hat{p}, \hat{t} \in \mathcal{IR}$, $i \in \mathcal{I}$, $a \in \mathcal{I} \cup \{?\}$, $z \in \mathcal{O} \cup \{?\}$, and $O \subseteq \mathcal{O}$. 

Each time that a user applies $a$ and observes $z$ and the time required by the transition verifies the restriction $\hat{p}$.

Some actions are performed and the time required verifies the time restriction $\hat{p}$.

If after performing some operations the user inputs $i$ then he must observe and output in $O$, the time required by this transaction satisfies the restriction $\hat{p}$ and the time for all the sequence satisfies the restriction $\hat{t}$.

Formal Definition of Invariants

$\phi$ is defined according to the following EBNF:

$$\phi ::= a/z/\hat{p}, \phi \mid \ast/\hat{p}, \phi' \mid i \mapsto O/\hat{p} \triangleright \hat{t}$$

$$\phi' ::= i/z/\hat{p}, \phi \mid i \mapsto O/\hat{p} \triangleright \hat{t}$$

$\hat{p}, \hat{t} \in \mathcal{IR}$, $i \in \mathcal{I}$, $a \in \mathcal{I} \cup \{?\}$, $z \in \mathcal{O} \cup \{?\}$, and $O \subseteq \mathcal{O}$. 
Invariant $\phi_1$

- If we see in the log the input `login` followed by `option_screen` and an amount of time included in $[20,40]$,...

$$\phi_1 = \text{login}/\text{option_screen}/[20,40],$$
Invariant $\phi_1$

- If we see in the log the input login followed by option_screen and an amount of time included in $[20,40]$, ...
- and we observe any sequence of input/output/time and then the input disconnection ...

$\phi_1 = \text{login}/\text{option_screen}/[20,40], \star/[0,\infty], \text{disconnection}$
Invariant $\phi_1$

- If we see in the log the input `login` followed by `option_screen` and an amount of time included in $[20,40],...$
- and we observe any sequence of input/output/time and then the input `disconnection` ...
- ... then if we observe `disconnection` we have to see the output `welcome_screen` in an amount of time belonging to $[10, 20]$.

$$\phi_1 = \text{login}/\text{option_screen}/[20,40],\quad \ast/[0,\infty],\ \text{disconnection}$$
$$\hookrightarrow \{\text{welcome_screen}\}/[10,20]$$
Invariant $\phi_1$

- If we see in the log the input login followed by option_screen and an amount of time included in [20,40],...
- and we observe any sequence of input/output/time and then the input disconnection ...
- ... then if we observe disconnection we have to see the output welcome_screen in an amount of time belonging to [10,20].
- ... In addition, the sum of all time values observed from the login until the welcome_screen output belongs to [35, $\infty$].

Invariant

$$\phi_1 = \text{login/option_screen/}[20,40],$$
$$\star/[0,\infty], \text{disconnection}$$
$$\mapsto \{\text{welcome_screen}\}/[10,20]$$
$$\triangleright [35,\infty]$$
Invariant $\phi_2$

- After observing any occurrence of login in the log,

$$\phi_2 = \text{login}$$
Invariant $\phi_2$

- After observing any occurrence of login in the log,
- then we have to observe option_screen or error_user.

$$\phi_2 = \text{login} \rightarrow \{\text{option\_screen, error\_user}\}$$
Invariant $\phi_2$

- After observing any occurrence of login in the log,
- then we have to observe option_screen or error_user.
- The amount of time associated with this input/output must belong to the time interval $[10, 40]$.

$$\phi_2 = \text{login} \mapsto \{\text{option_screen}, \text{error_user}\}/[10, 40] \uparrow [10, 40]$$
After inserting the last change into the profile_screen, the option screen and the time associated with these operations are included in the intervals \([10, 20]\) and \([20, 40]\) respectively. In addition, the total amount of time to perform this activity must belong to the interval \([35, 50]\).

\[
\phi_3 = \text{data/profile\_screen} \to \{\text{option screen}\} / \{20, 40\} \triangleq \{35, 50\}
\]
Invariant $\phi_3$

- After inserting the last change into the profile_screen,
- and after saving the current state of the system, the option_screen

$$\phi_3 = \frac{\text{data/profile\_screen}}{,\text{save} \mapsto \{\text{option\_screen}\}}$$
Invariant $\phi_3$

- After inserting the last change into the `profile_screen`,
- and after saving the current state of the system, the `option_screen`
- and the time associated with these operations are included in the intervals $[10, 20]$ and $[20, 40]$ respectively.

$$\phi_3 = \text{data/profile_screen}/[10, 20], \text{save} \mapsto \{\text{option_screen}\}/[20, 40]$$
Invariant $\phi_3$

- After inserting the last change into the `profile_screen`, and after saving the current state of the system, the `option_screen` and the time associated with these operations are included in the intervals $[10, 20]$ and $[20, 40]$ respectively.
- In addition, the total amount of time to perform this activity must belong to the interval $[35, 50]$.

$$\phi_3 = \frac{\text{data}}{\text{profile\_screen}}/\left[10, 20\right], \text{save} \leftrightarrow \{\text{option\_screen}\}/\left[20, 40\right] \triangleright \left[35, 50\right]$$
When the student is in the `option_screen` (after any input),

\[
\phi_4 = \text{?/option\_screen/}[5,35],
\]
When the student is in the option_screen (after any input),
if she inserts the input interaction marks, then the marks_screen will appear within 20 and 40 time units.

\[ \phi_4 = \text{option_screen}/[5,35], \text{marks} \mapsto \{\text{marks_screen}\}/[20,40] \]
When the student is in the `option_screen` (after any input), if she inserts the input interaction `marks`, then the `marks_screen` will appear within 20 and 40 time units. All the interaction must take between 30 and 70 time units.

\[
\phi_4 = \text{option_screen} / [5, 35], \quad \text{marks} \leftrightarrow \{\text{marks_screen}\} / [20, 40] \triangleright [30, 70]
\]
Algorithm: Correctness of invariants

Input: \( M = (S, I, O, Tr, s_{in}), \phi = \{a_1, \ldots, a_{n-1}, i_n \mapsto \mathcal{M}\} \) // either \( a_k = i_k/o_k/F_k \) or \( a_k = * \)
Output: Bool.

\( I' \leftarrow I; S' \leftarrow S; j \leftarrow 1; S'' \leftarrow \emptyset; error \leftarrow false; \)

\[ \text{While } (j < n) \]
\[ \quad \text{If } (\text{Head}(I') = *) \]
\[ \quad \quad \text{While } (S' \neq \emptyset) \]
\[ \quad \quad \quad \text{Choose } s \in S' \quad S' \leftarrow S' \setminus \{s\} \]
\[ \quad \quad \quad S'' \leftarrow S'' \cup \text{afterInp}\{s, i_{j+1}\} \]
\[ \quad \text{else} \]
\[ \quad \quad \text{While } (S' \neq \emptyset) \]
\[ \quad \quad \quad \text{Choose } s_a \in S' \quad S' \leftarrow S' \setminus \{s_a\} \]
\[ \quad \quad \quad Tr' \leftarrow \text{afterCond}\{s_a, i, o\} \]
\[ \quad \quad \quad \text{While } (Tr' \neq \emptyset) \]
\[ \quad \quad \quad \quad \text{Choose } (s_a, s_b, i_j, o_j, F') \in Tr' \]
\[ \quad \quad \quad \quad Tr' \leftarrow Tr' \setminus \{(s_a, s_b, i_j, o_j, F')\} \]
\[ \quad \quad \quad \text{If } (F' = F_j) \text{ then } S'' \leftarrow S'' \cup \{s_b\} \]
\[ \]
\[ I' = \text{Tail}(I') \quad j \leftarrow j + 1; \quad S' \leftarrow S''; \quad S'' \leftarrow \emptyset; \]
\[ \text{If } (S' = \emptyset) \text{ error } \leftarrow true \]
\[ \text{While } (S' \neq \emptyset) \]
\[ \quad \text{Choose } s_a \in S' \quad S' \leftarrow S' \setminus \{s_a\} \quad Tr' \leftarrow \text{afterCond}\{s_a, i_n, ?\} \]
\[ \quad \text{While } (Tr' \neq \emptyset) \]
\[ \quad \quad \text{Choose } (s_a, s_b, i_n, o, F') \in Tr' \quad Tr' \leftarrow Tr' \setminus \{(s_a, s_b, i_n, o, F')\} \]
\[ \quad \quad \text{If } (\langle o, F' \rangle \in \mathcal{M}) \]
\[ \quad \quad \quad S'' \leftarrow S'' \cup \{s_b\} \]
\[ \quad \text{else } error \leftarrow true \]
\[ \]
\[ \text{If } (S'' = \emptyset) \text{ then } error \leftarrow true \]
\[ \text{Return } (\neg error) \]
Global view, specification, invariant, implementation, and logs

**Specification**

```
φ₁ = login / option_screen / [20, 40], */ [0, ∞],
    disconnection → \{welcome_screen\} / [10, 20] ↦ [35, ∞]

φ₂ = login → \{option_screen, error_user\} / [10, 40] ↦ [10, 40]

φ₃ = data / profile_screen / [10, 20], save → \{option_screen\}
     / [20, 40] ↦ [35, 50]

φ₄ = ? / option_screen / [5, 35], marks → \{marks_screen\}
     / [20, 40] ↦ [30, 70]
```
Global view, specification, invariant, implementation, and logs

**Specification**

- Specification diagram showing states and transitions.
- States labeled with operations such as `login`, `option_screen`, `error_user`, `screen`, `profile`, and `marks`.
- Transitions between states with time constraints.

**Invariants**

- \( \phi_1 = \text{login}/\text{option_screen}/[20,40], */[0, \infty], \) disconnection \( \mapsto \text{welcome_screen} / [10, 20] \Rightarrow [35, \infty] \)
- \( \phi_2 = \text{login} \mapsto \{\text{option_screen, error_user}\}/[10, 40] \Rightarrow [10, 40] \)
- \( \phi_3 = \text{data}/\text{profile_screen}/[10, 20], \text{save} \mapsto \{\text{option_screen}\}/[20, 40] \Rightarrow [35, 50] \)
- \( \phi_4 = \text{option_screen}/[5, 35], \text{marks} \mapsto \{\text{marks_screen}\}/[20, 40] \Rightarrow [30, 70] \)

**Implementation**

- Implementation diagram showing a cube with states labeled 'a' and 'b' and an 'Out' arrow.

**Traces**

- Traces snippet showing code execution with tags like `<...>` and `<...>`.
- Code lines indicating variable assignments and function calls.

The PASsive TE sting tool and Osmius

Las Navas del Marqués, July 2009
Global view, specification, invariant, implementation, and logs

Do not have a full specification
- Partial specification.
- Only interested in some properties.

Invariants
\[ \phi_1 = \text{login}/[20, 40], \star/[0, \infty], \]
\[ \text{disconnection} \mapsto \{\text{welcome_screen}/[10, 20] \mapsto [35, \infty]\} \]
\[ \phi_2 = \text{login} \mapsto \{\text{option_screen, error_user}/[10, 40] \mapsto [10, 40]\} \]
\[ \phi_3 = \text{data/profile_screen}/[10, 20], \text{save} \mapsto \{\text{option_screen}\}/[20, 40] \mapsto [35, 50]\] \]
\[ \phi_4 = ?/\text{option_screen}/[5, 35], \text{marks} \mapsto \{\text{marks_screen}\}/[20, 40] \mapsto [30, 70]\] \]

Implementation

Traces
Since we assume that we have \textit{TFSM}, logs will follow this pattern:

\begin{align*}
\text{Input}_1/ & \quad \text{Output}_1/ \quad \text{Time}_1, \\
\text{Input}_2/ & \quad \text{Output}_2/ \quad \text{Time}_2, \\
\ldots & \\
\text{Input}_n/ & \quad \text{Output}_n/ \quad \text{Time}_n
\end{align*}
Theorem

- Let $S$ (a specification) and $I$ (an implementation) be two Timed Finite State Machines,
- Let $\phi$ be a correct timed invariant with respect to $S$.
- Let $e$ be a log recorded from $I$.

If the invariant $\phi$ does not match $e$ then $I$ does not conform to $S$. 

Algorithm: Correctness of traces with respect to invariants

```plaintext
input: $e = (i_1/o_1/t_1, \ldots, i_r/o_r/t_r) \in \log$, 
$\phi = \{ \xi_1/p_1, \ldots, \xi_n/p_n, i_f \rightarrow O/p_f \Delta \tilde{q}_f \} \in \Phi$
// where for all $1 \leq i \leq n$ we have that $p_i \in I$, 
// and either $\xi_k = i_k/o_k$, with $i_k \in I \cup \{?\}$
// and $o_k \in O \cup \{?\}$, or $\xi_k = ?$
// $i_f \in I$, $O \subseteq O$, and $p_f, \tilde{q}_f \in I$. 
output: $\text{bool}$

Struct $A \{ t_0 :: \mathbb{R}; t_e :: I; t_a :: \mathbb{R};$
  $\text{wild} :: \text{bool}; \phi_{aux} :: \Phi; \}$
$b :: \text{Stack}[A];$
b_{aux} :: \text{Stack}[A];$
token :: $A;$
error $\leftarrow \text{false}; j \leftarrow 1;$
while ($j \neq \text{length}(e) \land \neg \text{error}$) do 
  // we access the $j$-position of the log
  $(i/o/t) \leftarrow e[j];$
  $j \leftarrow j + 1;$
  token.$t_e \leftarrow 0;$
  token.$t_a \leftarrow [0, 0];$
  token.$t_a \leftarrow 0;$
  token.wild $\leftarrow \text{false};$
  token.$\phi_{aux} \leftarrow \phi;$
  aux $\leftarrow \text{treated}((i / o / t).\text{token}.\text{error});$
  // check if current position holds with
  // the first component of the invariant
  if (aux $\neq \text{null})$ then 
    push(b_{aux}, aux);
  while $(\neg \text{isEmpty}(b))$ do
    token $\leftarrow \text{top}(b);$ 
    aux $\leftarrow \text{treated}((i / o / t).\text{token}.\text{error});$
    if (aux $\neq \text{null})$ then 
      push(b_{aux}, aux);
    $b \leftarrow b_{aux};$
  return(\neg \text{error});
```
Core of PasTe

Invariants

File

Traces

Correct trace?
Core of PasTe

- Spec
- Correct spec?
- Invariants
- File
- Traces
- Correct trace?
Classification of different Invariant Suites

Invariant Suite 2

\( \phi_2 = \text{login} \mapsto \{\text{option_screen, error_user}\} / [10, 40] \uparrow [10, 40] \)

\( \phi_3 = \text{data/profile_screen} / [10, 20], \text{save} \mapsto \{\text{option_screen}\} / [20, 40] \uparrow [35, 50] \)

Invariant Suite 1

\( \phi_3 = \text{data/profile_screen} / [10, 20], \text{save} \mapsto \{\text{option_screen}\} / [20, 40] \uparrow [35, 50] \)

\( \phi_4 = ?/\text{option_screen} / [5, 35], \text{marks} \mapsto \{\text{marks_screen}\} / [20, 40] \uparrow [30, 70] \)

Question

Which one do we prefer? Invariant Suite 1 or Invariant Suite 2?
Classification of different Invariant Suites

Specification

Invariant Suite 2

\[ \phi_2 = \text{login} \mapsto \text{option_screen, error_user}/[10, 40] \triangleright [10, 40] \]

\[ \phi_3 = \text{data/profile_screen}/[10, 20], \text{save} \mapsto \text{option_screen}/[20, 40] \triangleright [35, 50] \]

Invariant Suite 1

\[ \phi_3 = \text{data/profile_screen}/[10, 20], \text{save} \mapsto \text{option_screen}/[20, 40] \triangleright [35, 50] \]

\[ \phi_4 = \text{/option_screen}/[5, 35], \text{marks} \mapsto \text{marks_screen}/[20, 40] \triangleright [30, 70] \]

Question

Which one do we prefer? Invariant Suite 1 or Invariant Suite 2?
Mutants: Evaluating invariants I

- Mutants are produced from the specification.
- We generate traces from mutants.
- An invariant suite is better if it *kills* more mutants.
Mutants: Evaluating invariants II

**Specification**

Applying Mutation Operators

**Mutant 1**

**Mutant n**
Mutants: Evaluating invariants III

Changing an output

The PASsive TEsting tool and Osnius
Las Navas del Marqués, July 2009
Changing the target state in a transition

The PASsive TEsting tool and Osmius

Las Navas del Marqués, July 2009
Altering the time value

Mutants: Evaluating invariants V
Core of PasTe

Spec \rightarrow Correct spec? \rightarrow Invariants

Spec \rightarrow File

File \rightarrow Correct trace?

File \rightarrow Traces

Correct spec?

Correct trace?
Core of PasTe

Spec → Correct spec? → Invariants

Spec → Mutants

generate

Mutants → File

extract

File → Correct trace?

File → Traces
Integration into Osmius

- Impl. traces
- PasTe daemon
- Log file
- Osmius Log Agent
- Master agent
Running Example I

- Mutant of SSadmin specification executing in *real time*.
- Paste daemon detecting periodically errors in the trace of the mutant.
Running Example II

Specification

TFSM

q0

q0 CONNECT WELCOMESCREEN 10 q1
q1 LOGIN ERRORUSER 30 q1
q1 LOGIN OPTIONSEREN 30 q2
q2 DISCONNECTION WELCOMESCREEN 10 q1
q2 PROFILE PROFILESCREEN 20 q3
q2 MARKS MARKSCREEN 30 q4
q2 REGISTER REGISTERSCREN 200 q5
q3 DATA PROFILESCREEN 10 q3
q3 SAVE OPTIONSEREN 30 q2
q3 RETURNOPTION OPTIONSEREN 10 q2
q3 RETURNOPTION NOTSPECTED 10 q2
q4 RETURNOPTION OPTIONSEREN 10 q2
q5 DATASUBJECT REGISTERSCREN 10 q6
q5 RETURNOPTION OPTIONSEREN 10 q2
q6 DATASUBJECT REGISTERSCREN 10 q6
q6 SAVEREGERISTRATION CONFIRMSCREN 80 q7
q6 SAVEREGERISTRATION NOCONFIRMSCREN 80 q7
q6 RETURNOPTION OPTIONSEREN 10 q2
q7 RETURNOPTION OPTIONSEREN 10 q2
Invariants

INVARIANTSI

LOGIN OPTIONSSCREEN [ 20 40 ] , * [ 0 -1 ] ,
DISCONNECTION - < WELCOMESCREEN > [ 10 20 ] - [ 35 -1 ]
LOGIN - < OPTIONSSCREEN , ERRORUSER > [ 10 40 ] - [ 10 40 ]
DATA PROFILESSCREEN [ 10 20 ] , SAVE - < OPTIONSSCREEN > [ 20 40 ] - [ 35 50 ]
? OPTIONSSCREEN [ 05 35 ] , MARKS - < MARKSCREEN > [ 20 40 ] - [ 30 70 ]
Running Example IV

The PASsive TEsting tool and Osmius

Las Navas del Marqués, July 2009
Current work

Planned applications

Planned applications


- Intrusion detection system (IDS). Invariants to express behavioral patterns of crackers, malware and/or disgruntled employees.
Methods to extract a set of invariants

- If we have a complete formal specification we have to check the correctness of the proposed invariant suite.
- If we do not have a complete formal specification we assume that the invariant suite is correct.
We automatically derive invariants from the specification… but there are a lot!!
We automatically derive invariants from the specification.

We combine this technique with other formal approaches like users models in order to obtain a relevant set of behaviours.
Develop a *native* Osmius agent
Develop a *native* Osmius agent

- Impl. traces
- Osmius Paste Agent
- Master agent
References

- Osmius http://www.osmius.net
- Paste http://kimba.mat.ucm.es/paste/
Publications


amn09d  C. Andrés, M.G. Merayo, and M. Núñez. Supporting the extraction of timed properties for passive testing by using probabilistic user models. In Submitted to QSIC’09, 2009.


Thank you for your attention!